Forensic applications of analog memory: the digital evidence bag

A review of information security and its applications in digital forensics, including the creation of a digital evidence bag by exploiting the properties of analog memory

Benjamin James Agnew

Bachelor of Engineering (Honours) (Electrical and Electronic)
University of Adelaide, 2015

A thesis presented for the degree of
Master of Philosophy
in
Electrical and Electronic Engineering
University of Adelaide

January 2022

Contents

Abstract

Digital evidence is electronic data that "has the potential to make the factual account of either party more probable or less probable than it would be without the evidence" [1]. We consider digital evidence stored on a physical memory device, collected in the field and transported to a lab where the digital content is stored and analyzed. Digital Forensics is the area of study that deals with the science behind this process, as well as establishing best practices and legal requirements. The core aspects of digital forensics are preserving evidence integrity and the chain of custody during the handling and storage of the evidence [2].

In this thesis, we look specifically at digital evidence where only digital data is collected (such as forensic photography), as opposed to digital evidence that also includes the storage medium (such as seized mobile phones). We review the existing procedures used for collecting and transporting evidence and explore how these processes could be improved to better suit this kind of digital evidence.

The field of Information Security deals with providing confidentiality and integrity of data, along with authentication and non-repudiation of both data and entities [3]. This is a widely researched and well developed area with many commercial applications, the most well known being internet security. We review and categorize the existing technologies used in information security into four avenues of approach based upon the fundamental security concepts of each: cryptography, widely witnessed, hardware security and exploitation of manufacturing defects. Many information security systems incorporate several of these approaches which leads to the overall security of the system being improved.

The aims of Digital Forensics and Information Security are similar, however the processes and systems used are very different. This partly reflects that digital forensics is usually subject to a greater level of legal scrutiny, but it also highlights that there are potentially opportunities to improve the processes and systems used.

Hence we develop the concept of a "digital evidence bag" (DEB), a device for the secure transport of digital evidence that has the same requirements as physical evidence bags: tamper-evident, unforgeable and clean. To achieve these requirements through technological solutions, we look at technology used in Information Security along with traditional forensic processes and explore how they can be adapted to create a DEB.

Given the nature of digital data, it is easy to produce exact copies and edit the data without loss of quality. From a forensics point of view, this strips out a lot of the imperfections that are usually exploited in the traditional forensic processes. However the technology used to build digital memory is still inherently analog and has non-ideal characteristics, which are usually obfuscated in the digital application space. We show how these characteristics can be exploited to achieve the DEB requirements.

We explore how a digital fingerprint for conventional digital memory could be used to meet the requirements of the DEB. We also propose a DEB based on analog memory cells which offers a novel method to meet the requirements.

Declaration

I certify that this work contains no material which has been accepted for the award of any other degree or diploma in my name in any university or other tertiary institution and, to the best of my knowledge and belief, contains no material previously published or written by another person, except where due reference has been made in the text. In addition, I certify that no part of this work will, in the future, be used in a submission in my name for any other degree or diploma in any university or other tertiary institution without the prior approval of the University of Adelaide and where applicable, any partner institution responsible for the joint award of this degree.

I give permission for the digital version of my thesis to be made available on the web, via the University's digital research repository, the Library Search and also through web search engines, unless permission has been granted by the University to restrict access for a period of time.

I acknowledge the support I have received for my research through the provision of an Australian Government Research Training Program Scholarship.

Signed: ______________.______________
Benjamin James Agnew

Date: 25-01-2022

Acknowledgements

To my supervisors, Dr Matthew Sorell and Professor Michael Liebelt, thank you for the many hours of discussions and advice, without which this thesis would not have been written. I will always be grateful for the teaching, friendships and experiences you provided.

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Context

Forensic science has been defined as "the application of scientific methods to establish factual answers to legal problems" [2]. Digital forensics is a sub area of this looking at how computers, computer systems and digital data can be used to help establish factual answers to legal problems. Digital forensic evidence is digital data that "has the potential to make the factual account of either party more probable or less probable than it would be without the evidence" [1]. This term often includes the devices that contain the digital data as well, for example hard drives and mobile phones.

During an investigation involving digital forensic evidence, this digital evidence needs to be collected, transported, stored, analyzed and then potentially presented in court. Throughout this process, evidence integrity must be maintained which means protecting the digital evidence against tampering or deletion. Chain of custody is also an important concept in forensic investigations, which refers to keeping thorough records of all actions involving the evidence along with who was responsible for them [2].

In this thesis we are looking at the transport and handling of digital evidence and how this is done securely: that is ensuring that evidence integrity and chain of custody is maintained throughout the investigation. We observe that the requirements of this process are similar to the requirements in the field of information security.

Information security is an area of study that looks at protecting the confidentiality, integrity and availability of data [9]. With the advent of the internet, this field has become more important than ever. Internet security (or cyber-security) is one of the most active and well researched applications of information security. The four core objectives in information security are confidentiality, data integrity, authentication, and non-repudiation [3].

1.2 Aim

We note that the aims and requirements in digital forensics are similar to those in the information security field. However after reviewing existing standards and procedures, we find that the processes used to achieve these aims are very different between the two fields. Digital forensics currently uses procedures, paper audit logs, physical security and other non-technical solutions to achieve these aims, whereas information security uses cryptography, widely witnessed data (blockchain), secure hardware and the exploiting of manufacturing defects to achieve the aims.

After reviewing the literature, we observed that the field of information security appears to be further progressed and more thoroughly researched than digital forensics, which leads to the following questions:

- Can we apply similar solutions from information security to the digital forensics field in order to better achieve the aims?
- And if so, what might these look like?

In investigating these questions, the concept of a Digital Evidence Bag (DEB) is introduced, a memory device for the secure transport of digital evidence. The background and requirements of a DEB are fully explained in chapter 2. Once the DEB has been defined, the questions were refined to be: Can we meet the requirements of the DEB by exploiting:

- the analog properties of digital memory, or
- the properties of analog memory cells?

1.3 Scope

In some cases, digital data is stored on a device where the device is also evidence itself. An example of this is computers and other electronic equipment seized from the crime scene. These devices contain digital evidence as well as other non-electronic evidence (fingerprints, DNA, etc.). In this thesis, we are not looking at these cases since the procedures used for the transport of physical evidence are well established.

We are looking at the cases where the storage device is not otherwise physically relevant to the investigation, only the digital data within it is relevant. Some examples of this include crime scene photography, CCTV footage and dashcam video. In these cases, the digital evidence can be copied, transferred, sent over the internet, etc., without any loss of evidence.

The DEB is aimed at the collection (and transport) phases of the forensic cycle. Processes for ensuring evidence integrity and chain of custody during the later phases are not in the scope of this thesis.

We found that there are four distinct avenues of approach used in information security technology: cryptography, widely witnessed (blockchain), secure hardware and the exploiting of manufacturing defects. Any one of these could potentially be used to address the requirements of the DEB, and a combination of them is likely to give a more secure solution. However in this thesis we have chosen to focus on the exploiting of manufacturing defects since it is a relativity new area of active research. It also potentially offers some innovative solutions to the unforgability requirement of the DEB, one of the four requirements of the DEB (fully explained in section 2.5.2) that deals with preventing fake copies of the DEB being created. This is also the requirement that existing digital evidence handling procedures have struggled with to some extent.

1.4 Overview

A background of digital forensics along with a review of existing procedures is given in Chapter 2. The concept and requirements of a DEB are also introduced and explained. Chapter 3 explains and reviews the existing technology used in information security. We then categorized the existing technology into four avenues of approach based on how they provide security: cryptography, widely witnessed data (blockchain), secure hardware and the exploiting of manufacturing defects. Manufacturing defects form the basis of digital fingerprinting.

Chapter 4 then explores how a digital fingerprint could be applied to conventional digital memory in order to help create a DEB. This includes experimentation to create a method for digital fingerprinting older generation EPROM digital memory chips. Then in Chapter 5 we explore a design of a DEB using analog memory cells. This outlines a novel method for the storage of digital data in analog cells that helps meet the requirements of the DEB. Following on from this, Chapter 6 looks at using memristors as analog memory cells. Simulations and experiments are carried out to ascertain if memristors could be used to build the DEB described in Chapter 5.

Finally Chapter 7 provides a conclusion of the work done and highlights the contributions that have been made by this thesis.

Chapter 2

Digital Evidence, Forensics and Investigation

2.1 Introduction to Forensics

Forensics is the process of answering legal problems using scientific methods across all areas of science from biology to physics and engineering. Digital forensics refers to the application of computer science and electronic engineering in assisting to solve legal problems.

Digital evidence is electronic data that "has the potential to make the factual account of either party more probable or less probable than it would be without the evidence" [1]. This data can be stored and/or manipulated on a computer system or electronic device or transmitted by a communications system. It comes in a number of different forms such as CCTV video, crime scene photography, phone call records, server logs and seized equipment such as computers, phones and hard drives [2].

Forensics is usually considered in the context of criminal law where law enforcement agencies conduct the forensic investigation and then use the evidence collected to prove or disprove the guilt of someone suspected to have committed a crime. However, organizations, both public and private, can also carry out forensic investigations. Often these investigations are looking at an incident that has occurred at the organization where management would like to know more about why and how the incident occurred.

As an example in the digital forensics field, if the security of an organization's servers is breached and data is stolen, a forensic investigation would then be carried out to determine how the breach occurred and what was stolen. The outcome of the investigation is usually to make changes to prevent such an incident from occurring again but a forensic investigation can also improve public relations after an incident and identify who was responsible.

2.1.1 Relevance of the storage medium

Digital evidence can be found and collected from a number of locations including crime scenes, data centres, archives and network traffic. In most cases it is then transported to a digital forensic lab for examination and analysis.

Here we consider two categories of digital evidence, cases where the physical storage medium is also evidence and cases where the digital data is evidence alone. The first case is one often associated with digital forensics, where electronic devices are collected as evidence. For example, computers and phones found at a crime scene are also physical evidence since they are objects that may have fingerprints, DNA and other non-digital evidence on them. In addition, they also contain digital data within the memory of the device which may also be used as evidence.

The second case is less common and often overlooked by standards. In this case the device the data is stored on (or transmitted on) is not relevant to the investigation. For example, crime scene photography, where the data is first stored on a SD card and then transferred to a database. CCTV footage is also in this category as the data is first stored on a digital video recorder before the relevant files are extracted and transferred to the forensic investigator.

2.1.2 Evidence integrity and chain of custody

There are two fundamental principles when dealing with evidence, the first, evidence integrity, refers to preserving both the original evidence and any copies of it in their original form. It is vital that evidence integrity is maintained throughout the investigation. In the case of digital forensics, this means ensuring the data hasn't been changed or deleted, either accidentally or intentionally (maliciously) [2].

The second fundamental principle, chain of custody, refers to the documentation that needs to be maintained throughout the forensic process. This documentation includes records of all actions preformed during the collection, handling, analysis and disposal of the evidence, along with who was responsible for each action. This documentation is used to prove the integrity and authenticity of the evidence.

In some cases, the confidentially of evidence is also required, which we will discuss further in section 2.5.2.

2.2 The forensic process

The digital forensic process is a five step process that provides some structure to how digital forensic investigations are carried out. The five steps are: [2]

1. Identification

2. Collection
3. Examination
4. Analysis
5. Presentation

Throughout the forensic process, evidence integrity and chain of custody must be maintained. The systems used for this are discussed further in section 2.3. The forensic process is also an iterative process in addition to the five linear steps. The collection/examination/analysis process is repeated as the results found in analysis can change the hypothesis or suggest other evidence that could be relevant to the investigation. Hence this process is then repeated with the collection, examination and analysis of the new evidence.

2.2.1 Identification

Upon arriving at the scene of an incident, an investigator needs to first identify any evidence (or potential evidence) that may be relevant to the investigation. This includes making an initial hypothesis to help guide in identifying what evidence to focus on. In criminal cases, this will often occur at a crime scene where the digital evidence is identified alongside traditional physical evidence. In other cases, the digital evidence may be identified in a number of locations including live data centres and archives.

Digital evidence may also be found in places other than the scene of an incident. For example, data stored with a cloud service provider can be evidence.

Before an incident even occurs, investigators need to be well trained and prepared so they can work quickly and methodically once at the scene of an incident. Some digital evidence may be in short-term, or volatile memory (eg. RAM) and needs to be dealt with quickly before it is lost. Investigators also need to be careful not to accidentally damage or erase any potentially useful data.

2.2.2 Collection

In the context of digital evidence, collection refers to the process of creating a digital copy of the data using forensically sounds techniques. All examination and analysis work should always be carried out using a copy and not the original data. This is done to minimize any chance of accidental changes to the original data.

The difficulty of collection depends largely on the device the data is copied from, along with the format the data is in. Devices such as hard drives, CDs, and USB memory sticks are usually relatively simple to copy. A write-blocker is commonly used when copying from these devices in order to prevent any accidentally changes to the data on the device.

Write-blockers are a hardware device that are placed between the computer and the storage medium and which are designed to only allow read operations to pass through from the computer to storage medium. Data contained within a mobile phone can be more difficult to copy and usually involves interaction with the Operating System (OS) on the phone in order to get access to the stored data.

Embedded devices often contain data traces which can potentially become digital evidence. The ease of copying this data can vary greatly depending on the device. In the more difficult cases, it many be necessary to physically remove the memory IC chip from the embedded device (a process know as “chip-off”) before placing it in a purpose built circuit for reading and copying.

Digital evidence can also be found in volatile memory such as RAM and CPU caches. Copying data from these locations needs to be done with care since the data in memory can change rapidly and can be completely erased by a loss of power. Encryption can also cause difficulties in collecting digital evidence. While encrypted data can still be copied, it is useless for further analysis unless the decryption key is known or can be found.

2.2.3 Examination

The examination phase involves looking through all the data collected and identifying any potential digital evidence. This involves restructuring, phrasing and preprocessing raw data in order to make it easier to analyze later.

In many cases, the raw data takes the form of a file structure and the examiner is looking to find files within it that may be relevant to the investigation. This may involve recovering deleted and partially damaged files as well as searching for hidden files. There are also often many files that are known to be irrelevant to the investigation and can be excluded, for example OS files.

The quantity of data collected in modern investigations is large and getting larger, driven by the ever increasing size of hard drives and other memory devices. As a result, forensic examiners need to be able to quickly identify where the most relevant data will be found. This process is known as “triage” where data is prioritized such that the most relevant data is examined first.

Since examination is a relatively repetitive task, there are a number of software tools designed to assist the examiner in quickly finding the relevant data. These software packages can search vast quantities of data, included compressed data, for strings, image matches, timestamps and cryptographic hashes of files. They include databases of known files to search for, such as OS files and other common software files that can be discounted from the investigation and known malware and contraband images that are likely useful to the investigation.

2.2.4 Analysis

In the analysis phase the data identified in examination is processed to establish facts relevant to the investigation. This includes evaluating the significance of the digital evidence and the person(s) responsible for it. This phase often involves visualization, timelining and linking pieces of evidence together in order to fully understand where each piece of evidence fits in the investigation.

The digital data can represent a diverse range of records including text, images, audio, databases and metadata. Each layer of the computer system, along with different software packages, will interpret these data objects differently. Analysis at the file system level will reveal timestamps and file metadata but will often miss metadata that is part of the file itself, for example EXIF data within an image file. EXIF (Exchangeable image file format) is a standard for encoded metadata such as camera settings and the time/location the photo was taken, all within the image file [10]. Hence it is useful to visualize each data object in a way that all aspects of it can be easily interpreted and analyzed.

Timelining involves sorting forensic data, based on timestamps, into chronological order. This allows the investigator to understand the order in which events occurred and may help form hypotheses on why and how they occurred. Timelines constructed from traditional forensic evidence can also be merged with digital forensic timelines to help understand the chain of events. This can then be used to explain how the digital forensic evidence applies to real-world events.

Link analysis is used in all forensic investigations, however it is particularly useful in digital forensic investigations. It involves linking similar or related pieces of evidence together and then visualizing all the links to help identify the relevance of evidence along with any missing links or missing evidence.

2.2.5 Presentation

The final phase involves documenting the findings and results of the investigation for presentation to a court of law or other audience. Whilst this report needs to thoroughly cover all aspects of the investigation, it also needs to be written for a non-specialist audience such as judges, jury members or company executives.

The report needs to cover all aspects of the investigation including, but not limited to, details of all evidence collected, examined and analyzed, roles and tasks of investigators and any information needed to reproduce the results of the analysis. In order to make reports easier to understand, diagrams, images and screenshots are often included to help explain processes used in the investigation. The report also needs to address evidence integrity and chain of custody requirements and outline how these have been maintained during the investigation.

2.3 Existing standards

There are a number of standards applicable to digital forensics produced by forensic science and policing organizations around the world. This section explores each of these in detail.

2.3.1 SWGDE

The *Scientific Working Group on Digital Evidence Framework* (SWGDE) has created a framework for a Quality Management System (QMS) for forensic science service providers [11]. It outlines a number of polices and procedures that should be in place for digital evidence handling and investigation. This covers a number of areas including staff qualifications and training, laboratory tools and equipment and periodic reviews of work done.

However there are few details about what these policies and procedures should actually contain. In the area of evidence handling, it simply states that there should be procedures for "*properly identifying derivative evidence, securing evidence and logging evidence interaction/transfer*". It also states that there should be procedures to enable access control for evidence storage. While there are no details on how these procedures should work, it seems to suggest that audit logs and access control (physical or software) are sufficient for ensuring evidence integrity.

2.3.2 Global Guidelines for Digital Forensics Laboratories

The International Criminal Police Organization (Interpol) has published the *Global Guidelines for Digital Forensics Laboratories* [12] which provides guidance on how to build and operate a digital forensics laboratory. This builds upon, and provides more detail around the framework set out by SWGDE.

After receiving the digital evidence, the examiner(s) at the digital forensics laboratory will analyse the evidence. The examiner(s) will then prepare a report before returning the evidence to law enforcement. While the evidence is in the hands of the digital forensics laboratory, it needs to be stored securely whilst it is not being worked on by the examiner(s). To achieve this, Interpol recommends that the digital forensics laboratory be built with surveillance systems and physical access control over rooms and/or cabinets designated for evidence storage. Off-site backups and archive systems are recommended to safe guard against equipment failure.

Guidelines are also provided for staff recruitment and training within a digital forensics laboratory. A security clearance should be required for all staff, along with a background check before they are hired. Due to the complexity and fast-changing nature of digital evidence, ongoing training for staff is recommended. Staff should be well trained in best-practices for digital evidence handling so as to minimize the chance of accidental damage to the evidence. During investigations, staff will require access to numerous tools, software and hardware, for which suitable training should be provided.

The Interpol guidelines also cover the basics of how forensic analysis is undertaken for both computer equipment and mobile phone devices. This covers a number of common procedures including data extraction and hard drive imaging, data analysis and file reconstruction through to data acquisition on both live and powered-off systems. Data extraction involving interaction with low-level hardware such as JTAG and chip-off are also discussed.

Guidelines around the reporting and presentation of evidence are also given, with a focus on ensuring the admissibility of the electronic evidence. While the specifics vary by jurisdiction, five general requirements for admissibility are given: "*Authenticity, Completeness, Reliability, Convincing and Proportionality*". However no detail is given as to how these should be achieved and proven to a court. A quality assurance system is recommend though along with accreditation which does assist in fulfilling the requirements for admissibility of evidence.

It is important to note that these guidelines only addresses digital evidence where the physical device holding the digital data is also evidence itself (eg. cases where physical devices with memory are the 'evidence'). It does not address the case where the evidence is purely digital data.

2.3.3 ETSI TS 103 643

The European Telecommunications Standards Institute (ETSI) have published a standard titled "*Techniques for assurance of digital material used in legal proceedings*" [13] which introduces the concept of a digital evidence bag (DEB). It outlines a system that can be used to store and transform digital evidence in a way that meets the requirements for admissibility for evidence in court. In this document,"transform" refers to any function or algorithm that is used to process and analyze the data (for example, image enhancement and color correction). ETSI defines a Purely Digital Transformation (PDT) to be a digital function that is repeatable, deterministic, pre-specified, fail-safe and well-defined.

Basic DEB

Three different levels of DEB are described, with the user to choose the most suitable one depending on the security requirements of the case. The "Basic DEB" simply consists of system to record the time, location, organization, data type and other information about the evidence collected. A Globally Unique Identifier (GUID) is also assigned to each piece of digital evidence to prevent any miss-identification of similar pieces of evidence. If a PDT is applied to the digital evidence, the time, name/version of PDT software and any other information needed to recreate the PDT, along with a new GUID for the transformed data is all recorded in the system.

In effect, the Basic DEB is the electronic equivalent of the paper-based audit log process. It is designed to detect and prevent accidental errors, not malicious attacks. No details are given on how to protect the records from tampering by a malicious attacker.

DEB with hashing

This option builds upon the Basic DEB in that the records along with the digital evidence data itself is all used as the input to create a cryptographic hash (effectively an integrity check). This hash is then to be "*stored in a manner which provides assurance that it cannot be changed over time*" and two methods are provided to achieve this.

The simplest method to store the hash such that it can not be changed is to store the hash in a database/data centre with physical access control and other conventional data security procedures (see chapter 3). The other method is to submit it to a system where it is published to a wide audience who can witness the hash and verify that it has not been changed. This is the concept behind blockchains and is discussed further in section 3.4.

Both of these methods allow the DEB with hashing to offer some protection against malicious attackers. Once the hash has been created and securely stored, any attempt to tamper with the data will be detectable by simply recalculating the hash and checking that it matches the one in storage.

DEB with input assurance

The DEB with input assurance also builds upon the Basic DEB in that along with all the basic records, it also records the delivery protocol or technique used to input data into the DEB system. This is usually in the form of credentials of the person who inputted the data, along with how and when they inputted the data, which assigns responsibility for the data's authenticity to a given person.

DEB with input assurance ties the integrity of the data to the person responsible for it. This is useful for cases where the responsible person can testify (in court or elsewhere) that the data is true and correct. This is similar to, and fits in with, traditional evidence procedures in court, which is discussed further in section 2.6.

2.3.4 ISO/IEC 17025

ISO/IEC 17025, titled *"General requirements for the competence of testing and calibration laboratories"* [14] is a widely accepted standard for scientific laboratories around the world. It sets out the requirements for a scientific laboratory and the organization that operates it. This includes requirements in areas such as impartiality and confidentiality of results, laboratory management, technical staff training, equipment maintenance and calibration and ensuring the laboratory is adequately resourced. Many countries have national accreditation bodies that provide laboratory accreditation to the ISO/IEC 17025 standard. In Australia, this is the National Association of Testing Authorities [15].

ISO/IEC 17025 focuses on two main aspects of laboratory work: testing and calibration. Testing refers the process where a sample is provided to the laboratory for testing. Technical staff will then use the laboratory's resources to preform tests on the sample and then report the results (and potentially analysis) back to the customer. Calibration is the process where a measurement instrument of unknown accuracy is compared to a reference with a known accuracy. From this process, the accuracy of the measurement instrument can be calculated and known for future use of the measurement instrument.

Relevance to Digital Forensics

ISO/IEC 17025 accreditation is a mandatory requirement for digital forensic labs in the UK. However there is some doubt over how well this standard applies to digital forensic procedures with many experts in the field suggesting that it is not relevant [16]. Along with the high cost of accreditation, many experts suggest that the requirements of ISO/IEC 17025 simply does not fit the work that is carried out by digital forensic practitioners.

One of the requirements of ISO/IEC 17025 is that laboratory equipment be tested, calibrated and verified. However, the equipment used in digital forensics is usually software packages designed to search, analyze and visualize data. Requiring that every copy of the software (in the hands of many digital forensic labs) each be individually tested, calibrated and verified is large waste of resources since copies of software all behave in the same way. Potential tampering with the software is an issue that has already been dealt with by using digital signatures to sign the software [17] (discussed further in section 3.3.4).

2.3.5 Forensic Photography (ANZPAA)

One of the most common forms of digital evidence, forensic photography is the process in which a law enforcement officer creates a visual record of a crime scene, in the form of video or still photographs, with the aim to present them in court as evidence. In order to maintain evidence integrity and chain of custody, the Australia New Zealand Policing Advisory Agency (ANZPAA) have published guidelines for law enforcement agencies to use when collecting forensic photography [18].

In addition to explaining file formats, image processing and analysis techniques, and best practices for transportation and storage of digital memory devices, it also outlines procedures to help ensure evidence integrity and chain of custody. The core part of this is the audit trail which is a log, recorded manually by the investigators, of all actions taken in relation to the photographs.

The storage of forensic photographs is covered in detail. Transferring images to a secure database is recommended to be done as quickly as possible. This can be a data server with appropriate access control or a number of other data storage technologies. Consideration must be given to how long the data will be stored for and if the storage media will last in archival storage situations. Write-once memory, in particular write-once CDs, are recommended to protect evidence integrity. However, write-once CDs are now largely obsolete.

While reducing the time between the photograph being taken and its transfer to secure storage can reduce the chances of both accidental and malicious tampering, it doesn't eliminate it completely. The photo is still stored on a reusable memory card after it is taken, and until it makes it to the secure storage, it can easily be tampered with by whoever is in possession of the memory card. Photos can also be deleted (even accidentally) and there may be no proof they ever existed.

2.4 Other research

There has been some published research into the area of digital evidence integrity and chain of custody. Whilst these have been focused on very specific problems, they are discussed below and may be applicable to other problems.

2.4.1 Custom SD card with log file

One possible improvement to the forensic photography process is a modified memory card that contains a log file stored in write-once memory [19]. In this system the memory card logs all read, write and erase operations to the log file along with a cryptographic hash of the written data. Any attempt to maliciously alter or delete the photo data would be detectable by comparing the hashes in the write-once memory to the data stored in the main memory of the card.

However this does not protect against intercept during the legitimate read and write operations. The other security weakness is that the log file from the memory card is required during verification. This requires that the log file be transferred to the secure storage (which is another opportunity for malicious tampering) or the memory card used needs to be physically present during verification which is difficult given that memory cards are reused, can fail and can be accidentally destroyed or lost.

2.4.2 Watermarking

The use of a watermark has been considered in [20] along with a biometric photograph of the photographer's eyeball. This allows strong proof of who took the photo and what camera it was taken with to be embedded in the image file as a watermark and cryptographic hash. However there is little protection from manipulation during the process of transferring the image to secure storage. Manipulating an image, recalculating the hash and then altering the watermark to match is relatively easy to do for someone with the right tools.

2.5 Evidence Bags

Physical evidence refers to physical objects that can assist in establishing facts in legal enquiries. This can be due to the object itself, for example, where it came from, who owns it, how did it get there, can all prove useful clues to the investigation. In addition to this, residues on the item, such as fingerprints and DNA, can also assist the investigation. Physical evidence is collected (often from a crime scene) and then presented directly in court, or tested and analyzed to revel further information.

To secure the physical evidence, the objects are placed in purpose designed evidence bags for transport and storage. To prevent contamination, these evidence bags are designed to be single use only. They are designed to be tamper evident and are stamped with serial numbers to prevent swapping out tampered bags with new ones [21].

In order to securely transport digital evidence to the lab, we require the digital equivalent of an evidence bag. In an effort to utilize physical evidence bags for digital evidence, police officers have simply placed the memory device in a evidence bag for transport and storage. This makes sense in the case where the memory device itself is also physical evidence. However when the digital evidence is independent from the memory device, this process may not offer any security benefits.

2.5.1 Digital Evidence Bags

The idea of a "Digital Evidence Bag" (DEB) is a relatively novel concept to create the digital equivalent of a physical evidence bag. This device would be used for the secure transport and storage of digital evidence. While mentioned in *ETSI TS 103 643* [13], the term DEB is only used to roughly outline the process of transporting, storing and analyzing digital evidence, as oppose to a physical device used for carrying out this process.

2.5.2 Requirements for a digital evidence bag

There is a need for a secure digital evidence bag which can be used to collect and transport digital evidence from the crime scene. After investigating existing procedures for physical evidence and the requirements for physical evidence bags, we formed these requirements for a digital evidence bag:

- Tamper evident. Any attempt to alter or erase the data on the device would be detectable.

- Un-forgeable. Any attempt to swap out the device for a fake one should be detectable.
- Clean. Before digital evidence is loaded onto the device, it needs to be clean and not contaminated with any data from a previous use.
- Offline. During evidence collection and transport, there may not be any network connection available.

The requirement of confidentially can also be included sometimes depending on the case. As with physical evidence, this is usually achieved with physical access control, although cryptography can also be used to this achieve confidentially with digital evidence. Since both of these solutions are well researched and are straight-forward to apply to digital evidence, we have not looked into the confidentially requirement any further in this work.

Tamper Evident

This is the requirement most often associated with physical evidence bags. If anyone attempts to alter or destroy the evidence, the evidence container needs to clearly show that it has been tampered with. Any attempt to alter the evidence without being detected needs to be prevented. Physical evidence bags achieve this with a one-time seal. After the evidence has been collected and sealed in the bag, it can only be accessed by tearing or cutting the bag open which is a destructive and detectable act.

Unforgeable

A malicious attacker can get around the tamper evident properties of a physical evidence bag by removing the evidence from the original bag, tampering with it and then resealing it inside a new bag that looks identical to the original. To prevent this, we need to ensure that each bag is unique and that a forged copy of it cannot be feasibly created. Physical evidence bags achieve this by stamping each bag with a unique serial number that is also recorded in a database.

Clean

Before use, we need to ensure each evidence bag is clean and free from any contaminants. This is usually achieved be using only new bags that have been shipped and sorted in protective packaging. After a bag has been used, it is disposed of.

In the case of digital evidence container we need to ensure the memory components are completely erased and contain no trace of data from previous uses. Or, if the devices are low-cost, it may be desirable to dispose of them after a single use.

Offline

Evidence may be collected in places without network connections. Devices may also be powered off during transport. As such, a digital evidence container must still operate correctly and provide security whilst operating offline or without power.

2.6 Digital Evidence in Court

As with all evidence presented in court, digital evidence must comply with the *Evidence Act 1995* [22] to be admissible in court. For the digital evidence to be admissible, the witness who collected the evidence (often a police officer) must testify to the court that the evidence is authentic and has not been tampered with. The court can then question this witness to establish the witness' credibility and hence the credibility of their evidence. Audit logs created during the processing and storage of the digital evidence are also presented to the court to assist in establishing credibility.

This process is very time consuming for police officers, lawyers and judges since judges rely on these testimonies to determine the authenticity of the digital evidence. There are also cases where the digital evidence has been collected by someone other than a police officer, for example, CCTV footage from a camera operated by a property owner or company employee. In this case the property owner or employee must testify to the court which can be intimidating for these people who usually have no involvement or knowledge of the court processes [23].

There is also a motivation to improve the process to reduce the time courts spend authenticating evidence. Having a witness testify, reading their submissions and making a judgment on their evidence is a time consuming process. The widespread popularity of fake videos (so called "deep-fakes") has only made this issue worse. Every piece of digital evidence is now scrutinized heavily since creating fake digital evidence is now viewed as a easy thing to do. While experts can easily distinguish most fake digital evidence, doing so takes time and experts then need to explain to the court how they reached their conclusion which uses up more court time. Hence a quicker, more automated system for authenticating digital evidence in court would be beneficial.

2.7 Conclusion

The digital forensics field deals with the collection of digital evidence in a way that evidence integrity and chain of custody is maintained. This ensures the evidence is admissible in court and trustworthy when it is used to establish facts in legal problems. The forensic process of identification, collection, examination, analysis and presentation outlines how a digital forensic evidence is handled. Evidence integrity and chain of custody must be maintained throughout this process.

There are a number of standards providing procedures and technology to assist in maintaining evidence integrity and chain of custody. This largely relies on the investigators keeping good audit records of all actions done with the evidence. Secure physical storage is also recommended where ever possible. These are mostly the same procedures used for handling physical evidence which have been adapted for digital evidence. In the case where the device containing the digital evidence is also physical evidence, using physical evidence handling procedures seems to be the right solution.

However in cases where the digital evidence exists independently from the storage medium, the procedures have some inherent downsides and security flaws. There is also a lot of dependence on good record keeping by investigators which is both time consuming and prone to mistakes. Some technical measures are used such as write-once memory and storing cryptographic hashes of the data. These do improve security and reliability to some extent however there is still room for further improvement.

Hence we seek to refine the concept of a Digital Evidence Bag, a device for the secure transport and storage of digital evidence. The focus for this device is on use cases where the digital data exists independently from the storage medium, for example, forensic photography, CCTV video and dashcam footage. The four main requirements of the DEB are that it be tamper evident, unforgeable, clean and operational offline. Throughout the rest of this work we will explore how such a DEB might be designed.

Chapter 3

Infomation Security

3.1 Introduction

Information security is an industry and research field focused on protecting the confidentiality, integrity and availability of data (often called the "CIA Triad") [9]. While the use of information security dates back thousands of years, the development, and extensive use of the internet has caused it to become much more common and widespread. Internet security (or cyber-security) has become one of the main applications of information security and as a result, a lot of research has been focused on improving and evolving information security.

There are a number of commonalities between information security and digital forensics. While the procedures used can be quite different, many of the science issues underlying each of these disciplines have common origins and concepts. Throughout this chapter we explore the aims and technology used in information security and how these could be applied to the digital forensics field, in particular to the creation of a Digital Evidence Bag (DEB).

3.1.1 Aims of Information Security

The *Handbook of applied cryptography* [3] defines four core objectives of information security: confidentiality, data integrity, authentication, and non-repudiation. We will briefly define and describe each of these, then discern how each is (or is not) related to forensics.

Confidentiality

Confidentially, also known as secrecy, is one of the aims most commonly associated with information security. It involves making sure information is only available to those that are authorized to access it. There are a number of ways to achieve confidentially such as encryption, access control systems and physical protection.

Data Integrity

Data integrity refers to securing data from tampering by malicious attackers. Ensuring data integrity involves detecting (and/or preventing) any unauthorized alterations to the data. Such alterations may include editing, overwriting, insertion, substitution and deletion of data. It is important to note that ensuring data integrity includes preventing the deletion/destruction of the entire dataset.

Authentication

Authentication involves identification and ensuring that the identity presented hasn't been forged. There are two major classes of authentication: entity authentication; and data origin authentication. Entity authentication refers to ensuring the entity you are communicating with really is who they claim to be. This is usually a two way process: both parties in the communication need to identify and authenticate themselves (prove their identity) to the other party. Data origin authentication refers to identifying where the data came from and ensuring the source is correct and not forged. Data origin authentication is very closely tied to data integrity in that verifying the origin usually verifies the integrity as well.

Non-repudiation

Non-repudiation prevents an entity from denying previous actions or commitments. The term is often associated with the validity of a contract that an entity has signed. Non-repudiation prevents the signing entity from denying they ever signed the contract. This is closely related to the authentication problem. For example, a digital signature needs to be authenticated to come from a given entity and given the existence of a valid signature, the signing entity cannot plausibly deny that they created the signature. Non-repudiation can also apply to data where it is useful to prevent an entity from denying that they are the source of the data.

Relation to the objectives in Digital Forensics

Of these four core objectives, data origin authentication is the objective of most interest in digital forensics and the collection of digital evidence. However, providing data integrity and non-repudiation is also important. As mentioned in section 2.5.2, confidentially is also important is some digital forensic cases.

3.2 Avenues of approach

A variety of approaches have been considered for addressing information security (and to a lesser extent, digital evidence handling). After reviewing the field, we have categorized information security into 4 main avenues or approaches:

1. Cryptography (relies on a secret key being kept secret)
2. Widely witnessed (relies on a piece of information being witnessed by a large number of people)
3. Hardware/physical security (relies on hardware designed to secure information)
4. Manufacturing variances/defects (relies on variations between components due to variations in manufacturing)

3.3 Cryptography

Cryptography has been around for thousands of years and has mainly been used to address the confidentiality problem. Before computers, cryptography was applied to written text and carried out using pencil, paper and other mechanical devices such as the famous Enigma cipher machines [24]. However modern cryptography, applied to binary data can be used to address all aims (see section 3.1.1) of information security. Due to its versatility and long history it has been used in internet-focused cybersecurity for many decades. Hence a large amount of research work has been carried out on finding weaknesses and improving security.

Figure 3.1 shows a basic outline of the cryptography process which takes place between two parties (referred to as Alice and Bob) wishing to have secure communication between them. First, Alice puts the plaintext (the raw binary data to be secured) through a mathematical function (referred to as 'encryption') along with the 'encryption key' to produce the output, referred to as the 'ciphertext'. A good encryption algorithm will produce ciphertext that shows no relation in the input plaintext. Alice then sends the ciphertext to Bob over an insecure channel (a channel that can be eavesdropped on and even altered by attackers). Bob then puts the ciphertext he receives into another mathematical function (referred to as 'decryption'), along with the 'decryption key'. The output Bob gets from this function will be the same plaintext that Alice started with.

The encryption key and decryption key are also binary data, which must be kept secret from any attackers eavesdropping on the communication channel. There is an assumption that an attacker knows the functions used for encryption and decryption, hence the secrecy of the keys is the only thing preventing an attacker from decrypting the data. The keys must also be complex enough that an attacker cannot guess the key by successively trying all possible keys until the correct one is found (this is known as a 'brute-force attack').

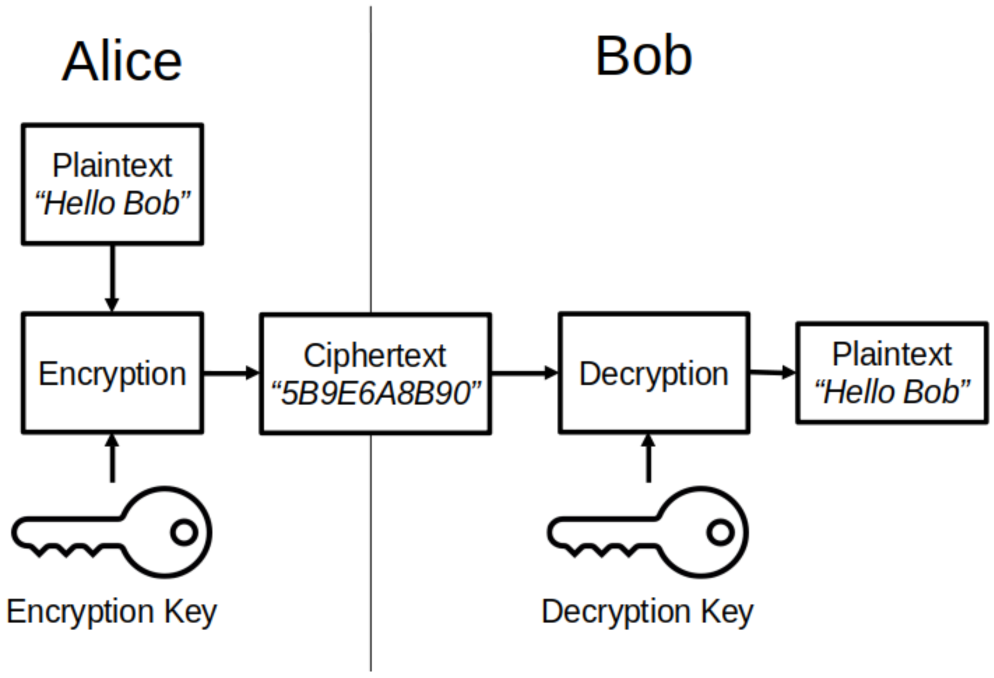


Figure 3.1: Block diagram of basic cryptography process

Man-in-the-Middle Attacks

A man-in-the-middle attack is a common form of attack on a communication system between two parties. The attacker intercepts the communication channel to either watch or tamper with the data that is been sent between the two parties. The attacker aims to do this in such a way that neither of the parties are aware the attack is happening. This is achieved by impersonating each party when talking to the other [25].

3.3.1 Cryptographic hashes

Cryptographic hashes are one of the main components of many information security systems, this section provides an overview of how they operate. A cryptographic hash is a mathematical function that can easily be computed in the forward direction but is impossible to compute in the reverse direction. So given some input data x, it is easy to calculate $y = H(x)$ where $H()$ is the cryptographic hash function. But given only y, there is no computationally tractable way to work out an x such that $H(x) = y$. Cryptographic hashes are sometimes referred to as 'one way functions' since they cannot be computed in reverse. In practice, the input data x is an array of bytes of any length (including zero) and the hash output y is fixed length integer usually expressed in hexadecimal format [26].

The properties of cryptographic hashes are:

- Deterministic. If $H(x) = y$ then this will always be true every time it is calculated for the same value of x
- Pre-image resistance. Given a hash value y, it is computationally infeasible to find an input x such that $H(x) = y$
- Second pre-image resistance. Given input data x_1, it is computationally infeasible to find a different input x_2 such that $H(x_1) = H(x_2)$
- Collision resistance. It is computationally infeasible to find two distinct values x_1 and x_2 such that $H(x_1) = H(x_2)$
- Avalanche effect. Even the smallest change in x will result in drastic changes in $H(x)$
- Non-inferability. Given a hash value $y = H(x)$, it should be impossible to infer anything useful about the data x that was used to create the hash

These properties make hashes useful in all areas of information security. For example, hashes can assist with data integrity by calculating and distributing the hash of the data requiring protecting. Anyone can then check the integrity of the data by recalculating the hash and checking that it matches the expected hash. This of course relies on the hash used to match against been secured against any tampering. There are many algorithms that as designed for cryptographic hashing, the most common ones been the Secure Hashing Algorithm family (SHA-1, SHA-2, SHA-3) [27].

3.3.2 Symmetric key cryptography

Symmetric key cryptography refers to the situation where both the encryption key and decryption key are the same. This single key is a 'shared secret' between Alice and Bob as shown in figure 3.2.

Symmetric key cryptography is the simplest and oldest form of cryptography however it is still used extensively in modern information technology. Examples of such algorithms are Data Encryption Standard (DES) and Advanced Encryption Standard (AES) which is used in SSL/TLS, WiFi, file system encryption and many other applications.

One of the difficulties of using symmetric key cryptography is that the key (the 'shared secret') needs to be known to both Alice and Bob. That is, it needs to be created and then shared via a secure channel so as to prevent an eavesdropping attacker from getting the key. This is a problem in many situations where there simply isn't any secure channel between Alice and Bob.

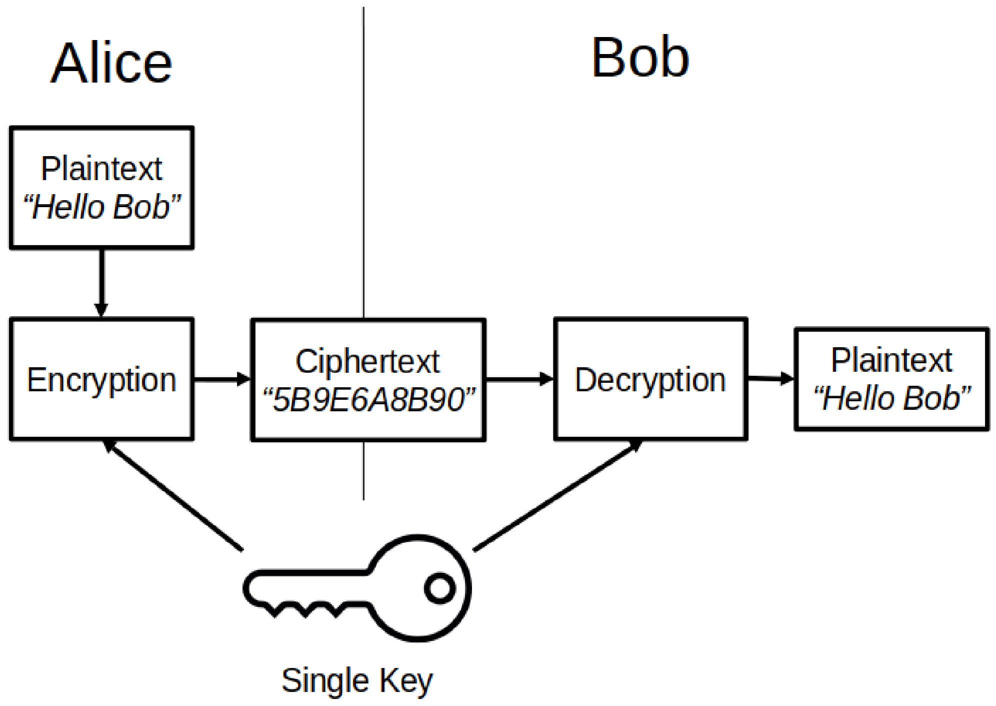


Figure 3.2: Block diagram of symmetric cryptography process

3.3.3 Public key cryptography

Public key cryptography is distinct in that is uses different keys for encryption and decryption as outlined in figure 3.3. These two keys are referred to as the 'public key' and 'private key' which together are known as a key pair. It is import to note that you cannot calculate the private key from knowing the public key [3].

Public key cryptography allows Bob to share his public key with Alice over an insecure channel since if an attacker has knowledge of the public key, it cannot be used to decrypt the ciphertext. Alice can use the public key to encrypt the message she wishes to send to Bob and then only Bob can decrypt it since he is the only one that knows the private key required for decryption. There are a number of algorithms used for public key cryptography such as Diffie-Hellman, RSA and Elliptic curve.

One of the weaknesses in public key cryptography is how to establish trust in the public key that Alice receives to encrypt the message. An attacker could easily send a public key and claim that it is 'Bob's public key' when in fact it is actually a public key that the attacker created. Figure 3.4 shows a basic man-in-the-middle attack in a public key encrypted message from Alice to Bob. When Bob goes to send his public key to Alice, the attacker blocks it and swaps it out for a fake public key belonging to the attacker. Since

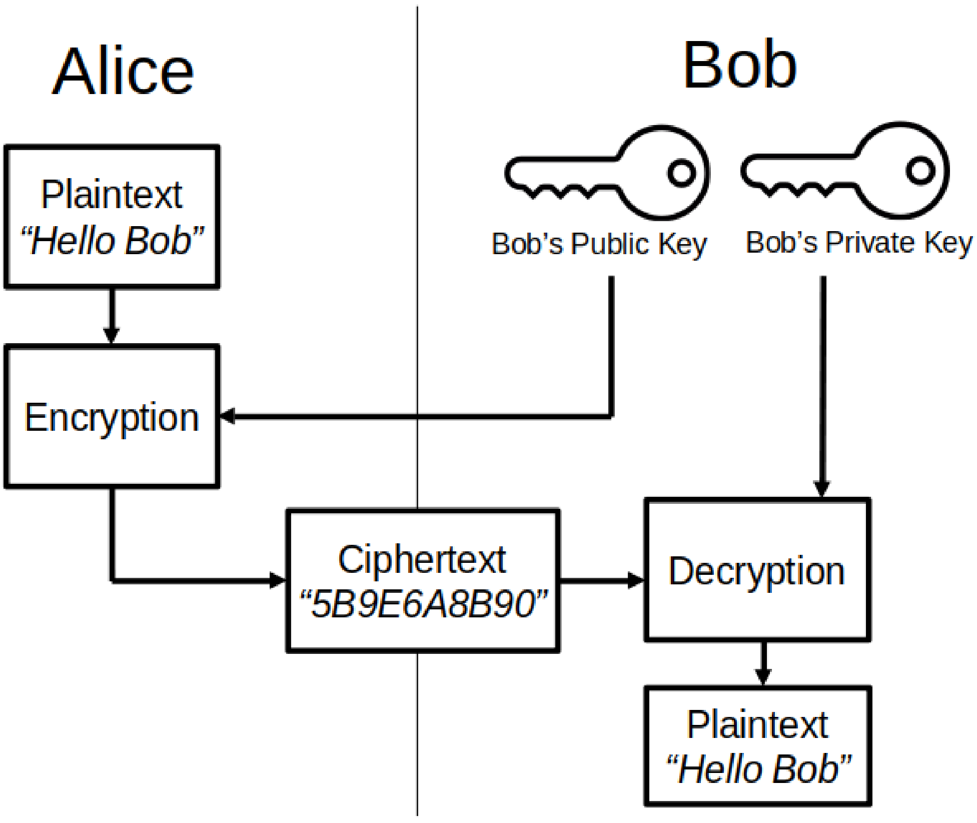


Figure 3.3: Block diagram of public key cryptography process

Alice has no way of verifying the public key she receives, she assumes it is Bob's and uses it for encryption. Since the attacker is in possession of the private key that goes with the fake public key, they are then able to decrypt the message Alice sends.

Many solutions to this problem have been put forward, the most well-known being the internet's public key infrastructure (PKI). The users (Alice) place their trust in a small number of central authorities, each known as a 'certificate authority' (CA). The job of the CAs is to verify that the public keys belong to who they say they are from. The large amount of trust in CAs has made CAs a prime target for attackers, the best example of this been the attack on the DigiNotar CA in 2011 [28].

Another approach is the "Web of Trust" system originally designed for the PGP email system. In this system, public keys are effectively shared via a trusted mutual friend (or multiple mutual friends). Upon receiving someone's public key, each user stores it and then offers to share it with their friends, similar to how mobile phone numbers are shared between friends (and friends of friends) [29].

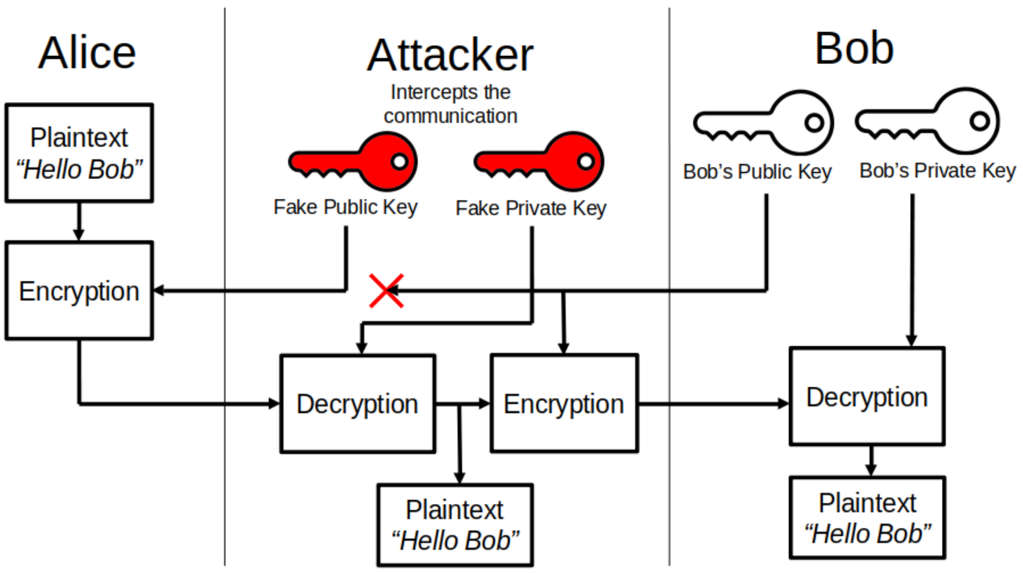


Figure 3.4: Block diagram of a man-in-the-middle attack on a message secured with public key cryptography

3.3.4 Digital signatures

So far we have only looked at the confidentiality problem which is what cryptography was originally designed to solve. However the processes used in public key cryptography can be 'reversed' to address the data integrity, authentication and non-repudiation objectives. In this situation, the private key is used for encryption and the public key is used for decryption as shown in figure 3.5. This process is referred to as a "digital signature" where the encryption algorithm is used to "sign" the data and the decryption algorithm is used to verify the signature [3].

For example, let's say that Alice wants to verify some data she received from Bob is free from tampering in transit. Bob encrypts the data with his private key ("signs" it) and then sends the ciphertext to Alice. Alice knows (and must trust) Bob's public key and uses it to decrypt the ciphertext she receives. An attacker, who also knows Bob's public key, can of course decrypt the ciphertext and view the data. However, if the attacker tries to tamper with the data, they will not be able to encrypt it since they don't know Bob's private key. By successfully decrypting the data with Bob's public key, Alice can be sure that it was encrypted with Bob's private key and hence can be sure that it came from Bob without any tampering along the way.

In practice, Bob first calculates the cryptographic hash of the data and then encrypts (signs) the hash as oppose to the actual data. Alice then calculates the hash of the data she received as well as decrypting the signed hash from Bob. If the two match then Alice can be

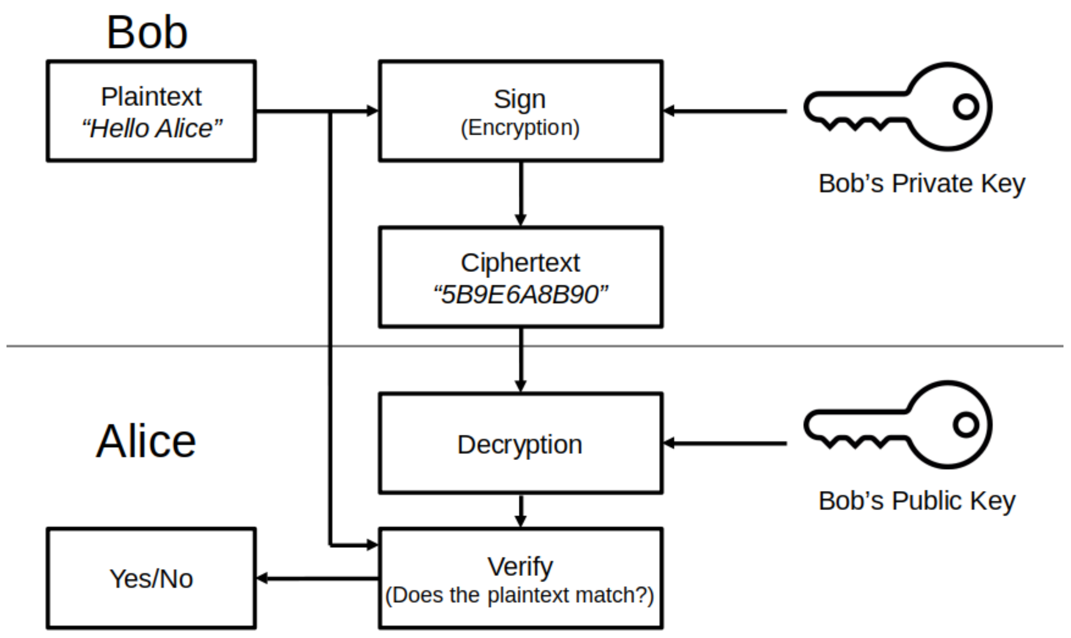


Figure 3.5: Block diagram of digital signature algorithm

sure the data hasn't been tampered with. By using hashes, the computational requirements for both Alice and Bob are reduced.

3.3.5 Authentication

In the authentication problem, Alice is talking to someone who claims to be Bob. She wishes to authenticate Bob which means that Bob needs to prove that he is really Bob. In many applications this is done with a password that is only known to both Alice and Bob however this relies on a secure channel to send the password between them. If they are communicating over an insecure channel, they need to use a challenge-response process to authenticate.

Challenge-response refers to the process where Alice sends a challenge to Bob, for example "What is your mother's maiden name?" Assuming that only Alice and Bob know the answer, Bob can respond with the answer back and Alice can verify if it is correct. This of course relies on the attacker not knowing the answer to the challenge which, if they are eavesdropping on the communication channel, they would after Bob has responded with it. To protect against this attack, it is important that Alice never sends the same challenge twice, that is, a new challenge is required every time she wishes to authenticate Bob.

In practice, public key cryptography can be used to create a challenge-response system as shown in figure 3.6. Alice selects a random number (that she has never used before) and sends it to Bob. Bob then encrypts this number with his private key and sends the encrypted number back to Alice. Since Alice knows (and trusts) the public key she has for Bob, she can decrypt it and compare the result it to the random number she sent. If they match, she can be sure that she is talking to Bob since only Bob is able to encrypt the number with his private key.

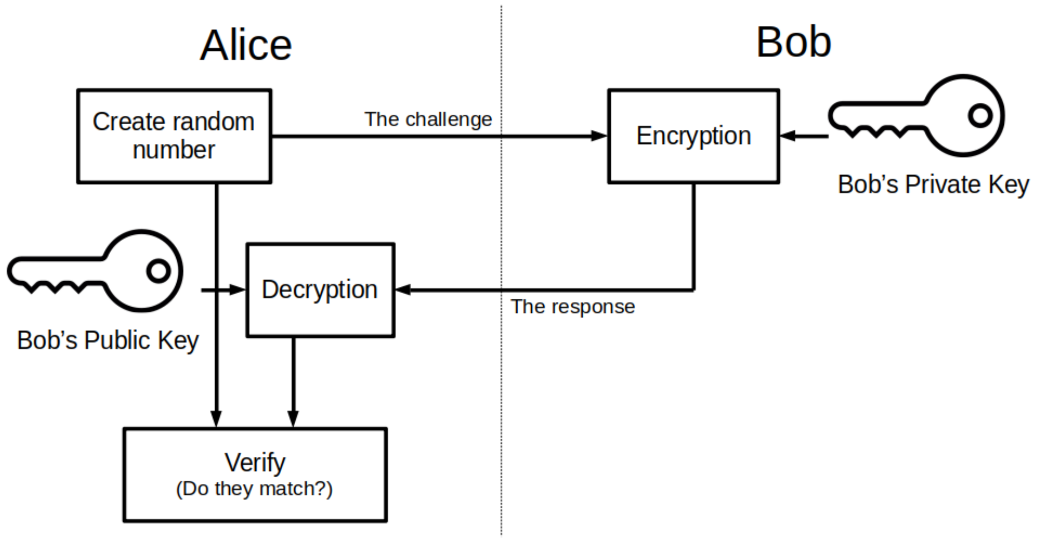


Figure 3.6: Block diagram of challenge-response algorithm

3.4 Widely witnessed

This idea relies on a simple concept: once something has been witnessed by a sufficiently large number of people, it is impossible to change it without someone noticing the change. For example, the final score of a televised and widely viewed football match cannot be tampered with by a malicious party since any alterations would immediately be noticed by the people that viewed the match. However extending this concept in a practical way in information security is a little complicated since data needs to be kept confidential and a suitably large number of witnesses needs to be found.

Given that we wish to keep the data confidential while ensuring data integrity via wide witnessing, we can't just show the data directly to the witnesses. So we first need to put the data through a cryptographic hash function and then get the hash value to be widely witnessed. Due to the properties of hash functions (see section 3.3.1), any tamping with the data would result in the hash value changing.

Widely witnessed is only useful for addressing the data integrity aim from section 3.1.1.

However by applying data integrity protection to things such as log files, it makes it very difficult for an attacker to cover their tracks.

3.4.1 Merkle trees

Merkle trees (also known as hash trees) allow for large amounts of data (potentially from different entities) to be protected by a single hash value. To do so each piece of data is hashed, then two (or more) hashes are combined and hashed again to create a single hash for both pieces of data. This process is repeated many times over as shown in figure 3.7 to create a tree structure with a single root hash protecting all pieces of data.

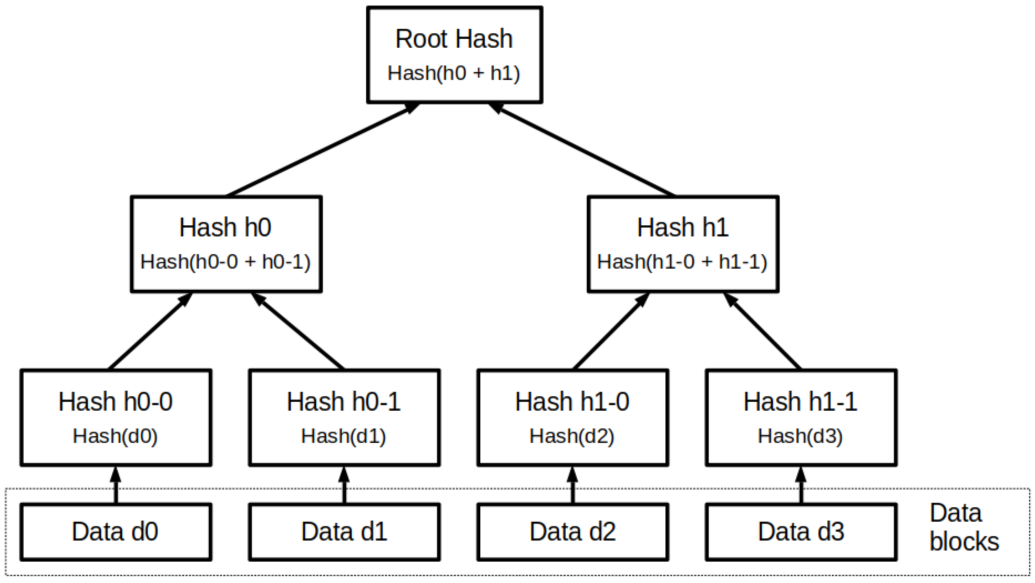


Figure 3.7: Block diagram of merkle tree structure (note that "+" refers to appending two strings together, not addition)

Figure 3.8 shows how this process would work with data from multiple entities. Each entity calculates their own 'sub-merkle tree' to get a root hash for all their data. The root hashes from each entity are then combined by one or more aggregators to create a global root hash. Protecting this root hash allows tampering in any of the data below it to be detected.

3.4.2 Hashchains

If we can secure the hash value then the data is protected from tampering. However, in the simple case, the attacker can easily re-calculate the hash value and over-write the original hash value. So to make this harder, we introduce the concept of hashchains. Assume we

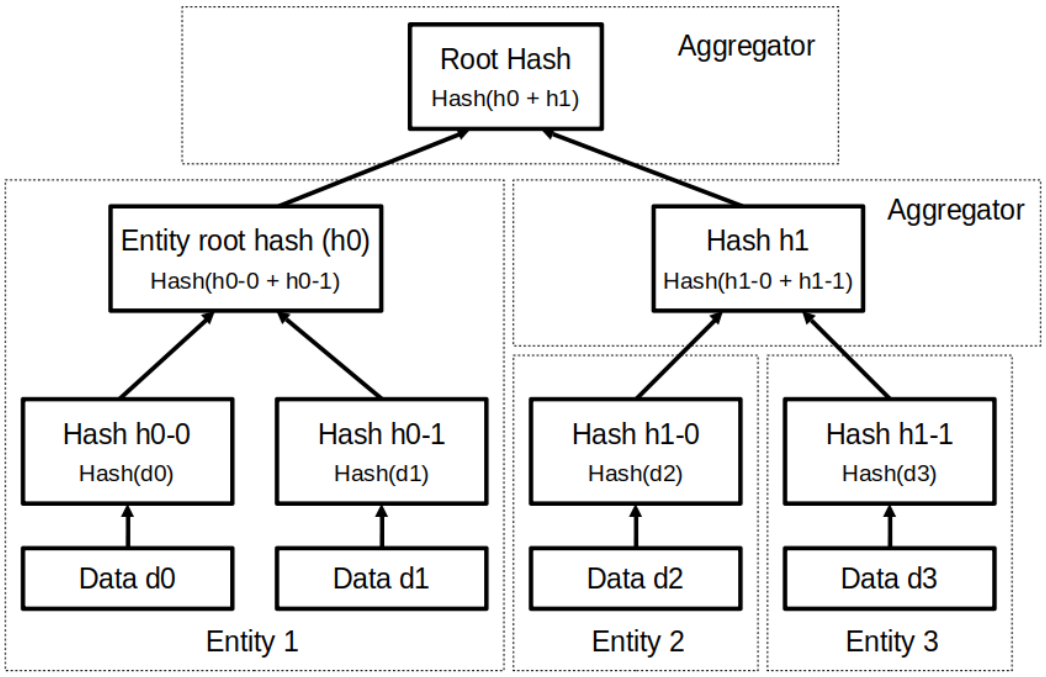


Figure 3.8: Merkle tree with with data from multiple entities (note that "+" refers to appending two strings together, not addition)

have a new hash value (which we wish to protect) arrive periodically with time interval t. Each time a new hash arrives, we append it to the top hash in the chain (from the previous time interval) and calculate the hash of this data plus some metadata such as a timestamp. This new hash becomes the new hash at the top of the chain. This process repeats at each time interval to create a chain as shown in Figure 3.9. In practice, the new hash for each time interval usually comes from the root hash of a Merkle tree.

In order for an attacker to tamper with the data and cover their tracks, they would need to recalculate every hash in the chain after the bit of data they tampered with.

3.4.3 Blockchains

Blockchains introduce the idea of 'widely witnessed' to hashchains to address the problem of an attacker recalculating the entire chain. If the top hash of the chain is widely witnessed at each time interval, it becomes impossible for an attacker to alter any of the hashes without one of the witnesses noticing.

The most well know form of blockchain is a public, decentralized blockchain. This system is made up of a large number of nodes (independent entities) communicating via a network (the internet) where a common standard for the blocks is agreed upon and used by all the

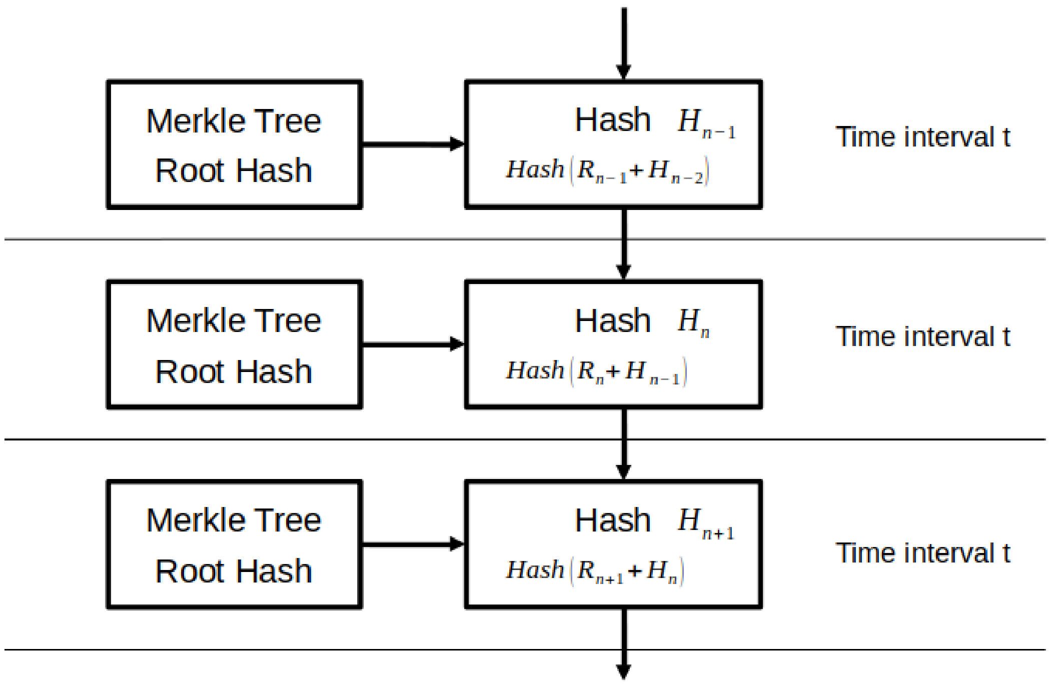


Figure 3.9: Block diagram of a hashchain (note that ”+” refers to appending two strings together, not addition)

nodes. Each block is data structure that contains the hash of the previous block, hashes of the data to be protected and other metadata. Assuming the block conforms to the standard, any node can send a block out to all other nodes on the network and the other nodes will witness the block, then attempt to use the block to create the next block in the chain. The nodes on the network compete with each other to create the next block in the chain, and once one of them creates the next block, they will race to get it witnessed by other nodes. As long as the nodes are independent such that no single entity has control over more than 50% of the network, then there is no way that a malicious attacker can tamper with the data undetected.

The most well-known example of a public, decentralized blockchain is the Bitcoin blockchain [30] which is mainly used to secure financial transaction records from tampering. In the Bitcoin system, nodes called “miners” compete to create the next block in the chain which are created approximately every 10 minutes. Each block contains the hash of the previous block, a timestamp, the root hash of a Merkle tree and other relevant data. The Merkle tree is made from all the financial transaction records for that block in the chain. Once a miner creates a valid block they share it with all other miners who witness the block and they get rewarded with new bitcoins for their effort to create the block.

It is also possible to run a centralized blockchain where a single entity is in control of the system. An example of this is the Keyless Signatures Infrastructure blockchain run by Guardtime which is aimed at securing data on commercial servers [31] [32]. At each time interval, the entity running the blockchain collects hashes from participants, calculates the Merkle tree and then the next block in the chain (which contains the Merkle tree root). The new blocks can then be secured by having them witnessed by the participants or by using cryptographic/hardware methods.

3.5 Hardware/Physical security

It is possible to design electronic hardware so that it fulfills some of the aims of information security. Depending on the aim and application, there are many different designs and methods used.

3.5.1 Write-once memory

Whilst this was not originally designed for security applications, write once memory has useful properties for ensuing data integrity. Once data has been written to the memory, an attacker is unable to tamper with the data since the physical design of the memory device prevents overwriting or deleting data. However it is still possible for an attacker to write the tampered data to a new memory device and then swap the original memory device out for the new one.

3.5.2 Securing microcontroller code

Microcontrollers are single integrated circuits (IC) that contain a CPU, program memory, data memory and various other modules all inside a single package. While they were originally designed as small, low cost computing platforms, it soon became apparent that the code stored in the program memory was a form of intellectual property and hence a target for attackers. To protect this, microcontrollers were built with special circuits that prevented the unauthorized reading of internal memory. To achieve this special memory cells (called security bits) are set after writing the program memory which disable the read out circuit. The microcontroller was designed such that the security bits could only be cleared by erasing the entire memory array. The only way an attacker could bypass the security bits was to depackage the IC and interact with the internal circuits which is expensive to do and requires specialist knowledge and equipment. [33]

3.5.3 Smart cards

The designs used to secure microcontroller code can be adapted for protecting other sensitive data such as cryptographic keys. Some examples include ID cards, bank cards and SIM cards. In addition to memory for the cryptographic keys, these cards contain hardware to implement cryptographic algorithms such as encryption, digital signatures and challenge-response. As a result, the secret keys never leave the card which makes it very hard for an attacker to recover the key if they are in possession of the card.

In addition to this, many card circuits require a PIN code to be entered correctly before the cryptographic hardware is enabled for use.

3.5.4 Hardware Security Modules

There are a number of products sold as Hardware Security Modules [34]. These take the concepts used in microcontrollors and smart cards then scale them up to a size suitable for data centre use. These devices are designed to store cryptographic keys, carry out the computation required for encryption/decryption and perform any other general computing requirements that a server may need to do. In addition to implementing access control on network/electrical interfaces, HSMs are also designed to be physically secure. Hence even if the HSM is physically stolen, the attackers are still unable to extract the secret keys from it. This is achieved by active sensor circuits that monitor things such as movement, temperature and power supply which then trigger an erasure of sensitive data if necessary [35]. In addition to this, the ICs within the HSM are designed using similar techniques used to secure microcontroller code, where the only way to extract the data from the ICs is to depackage the chips which requires expensive specialist equipment to do [36].

3.6 Manufacturing variances/defects

When electronic components are manufactured, there are tolerances in the specifications that the component must meet. This reflects the fact that it is very difficult to make two components that are identical in every way, so making them close enough is good enough. This concept is even more apparent in integrated circuits with millions of components such as digital memory. In these chips it is also expected that a certain percentage of the components simply won't work at all. But as long as this percentage is kept small enough, the failed components can be isolated and the chip will still function well enough for the end user. As a result of this, the per-unit cost can be kept very low for the end user.

For many years these variations between components have caused issues in circuit design and engineers have sought methods to minimize the effect of these variations. One of the many benefits of digital electronics is that it removes and ignores these variations as the digital signals travel though the logic gates. Hence, even if the transistors are all slightly different, it is possible to make two digital circuits that produce the exact same digital output. However these variations between components can be a source of information useful for security purposes (in particular, forensic purposes). With the shift to digital electronics and digital data, exploiting these variations for security purposes has become more complex.

This area is focused on the data integrity and authentication issues from section 3.1.1. This section provides a summery of some of the existing research into exploiting manufacturing variances and defects for security purposes.

3.6.1 Mathematics of Correlation

Once we have measured the manufacturing variances we need a method to compare the results. The sample correlation coefficient, a method to compare and quantify the similarities between arrays of data is used. This mathematical process outputs a number between -1 and +1 that represents how similar two arrays of data are [37].

If the correlation coefficient is close to 0 the arrays are said to be 'uncorrelated', meaning they have no similarities. For results further from 0 the arrays are said to be 'correlated', meaning there are similarities. For a positive correlation (closer to +1) the arrays look the same and for a negative correlation (closer to -1) the arrays look like a flipped version of each other.

To calculate the correlation between two arrays, labeled as X and Y below, we first 'normalize' the data by shifting and scaling it so it has a mean of 0 and a standard deviation of 1. Each element in X is then multiplied by the element Y in the corresponding position. The results of each element multiplication are them summed together and the final result divided by the number of elements in the array to produce a single number which represents the correlation between the two data arrays. The mathematics of this process is shown below.

Two arrays of data: $X(i)$, $Y(i)$ (both of equal length n)

Normalize: $X(i) = \frac{(X(i) - \text{mean}(X))}{\text{std-dev}(X)}$

(repeat the normalization for $Y(i)$)

Correlation: $C = \frac{\sum (X(i) \times Y(i))}{n}$

3.6.2 Digital Fingerprint

There is no clear definition of what a 'Digital fingerprint' is and the term has also been used for unrelated concepts in other fields. In this work, a digital fingerprint refers to a set of data with the following properties that can be used to identify an electronic device.

- It is obtained by real-world measurements on the device. It can't be calculated in simulation since it relies on measuring manufacturing variations.
- It is almost certainly unique to that device. Other devices from the same production line will have different digital fingerprints. The chances of two devices from the same production line having the same (or similar) digital fingerprint needs to be very small, for example less than 1 in a billion.
- It is difficult to forge. You cannot manufacture a device to match a given digital fingerprint or intentionally manufacture two devices with the same fingerprint.
- If the same device is measured twice, the digital fingerprints from each measurement should be the same (within the margins of measurement uncertainties, and under the same measurement conditions such as temperature and supply voltage)

In practice, a digital fingerprint is made up of measurements from a large number (>1 million) of components in the device. Each measurement is only a single scalar number representing some physical characteristic of that component. While individual component measurements are unlikely to be unique, once a large number of them are recorded in a set order, the resulting array of numbers is likely to be unique. Hence, this array is used as the digital fingerprint. Figure 3.10 shows a simple example of this with only six components, in practice a much larger number of components are used to ensure uniqueness.

Usually correlation is used to compare two fingerprints. If two fingerprints have a low correlation, it suggests that the fingerprints came from different devices. A high correlation suggests that fingerprints come from the same device. The values used for 'high' and 'low' vary depending on the application and are usually determined through controlled experiments.

3.6.3 Photography as an example of digital fingerprinting

The core component of every digital camera is the image sensor, which is made up of an array of photodiodes. Each photodiode creates one pixel of the image by converting the light intensity incident on that pixel of the senor array to an analogue voltage (which is then feed into an Analog to Digital Converter (ADC)). While manufacturers aim to make every photodiode the same, in practice there are variations between how each one reacts to light. There are also some photodiodes which simply don't work at all. [38]

It is possible to measure and exploit these manufacturing variations to create a digital fingerprint for the image sensor and hence for the camera. The variations in how each photodiode reacts to light are referred to as 'Photo response non-uniformity' (PRNU). Whilst they are small enough to be unnoticeable to the human eye in the single image, by taking

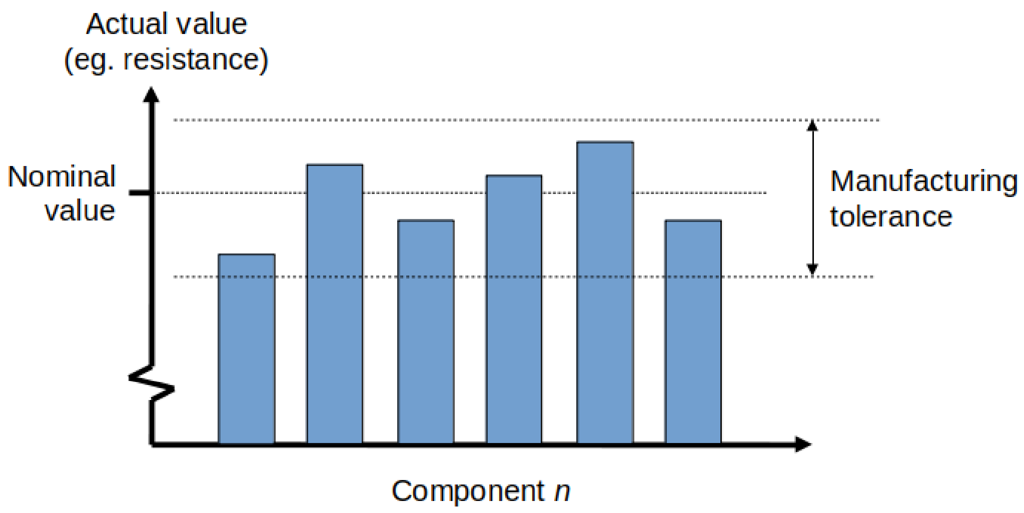


Figure 3.10: The small variations between components is used to create a unique digital fingerprint of the device

a large number of photos with the same camera it becomes possible to measure these variations. This process involves taking a large number of images of a ‘flat-field’ which is an image with an equal amount of light incident on every pixel. By averaging the resulting images, a digital fingerprint for the camera can be created.

3.6.4 Digital memory

There is some existing research on extracting forensic information from flash memory chips with the aim of creating a digital fingerprint for the chip. Flash memory cells are made from floating gate transistors and it is known that the charge needed on the floating gate to store a given value varies greatly from one cell to the next. In order to minimise the write-time for the user, the software monitors the writing process and finishes the operation as soon as enough charge to represent the correct binary value has been added to the floating gate. As a result, the write-time for each cell (or in practice, block) is different. By measuring the write time for every cell on the chip it is possible to create a digital fingerprint for the memory chip. This of course involves writing to every memory cell, which is a destructive process and can be a time consuming process as well. [39]

It is also possible to create a digital fingerprint by measuring the location and frequency of bit errors in the memory array. Certain floating gate transistors are more likely to cause a bit error than others and by measuring the error rate for each cell (or in practice, block), a digital fingerprint for the device can be created. The error rate is also influenced by external factors such as temperature, number of write cycles and storage time. Hence these variables need to be taken into account when measuring the bit error rate for a given memory chip. [40]

3.6.5 Physically unclonable functions

It is possible to create challenge-response authentication devices (see section 3.3.5) that use manufacturing variations to provide security. A simplified diagram of challenge-response authentication is shown in figure 3.11 which doesn't necessarily rely on secret keys and cryptography. As long as something about the function Bob uses is kept secret from the attackers, then they are unable to impersonate Bob. However Alice needs a method to verify that Bob's responses are correct so she needs to know some information about the secret function Bob uses.

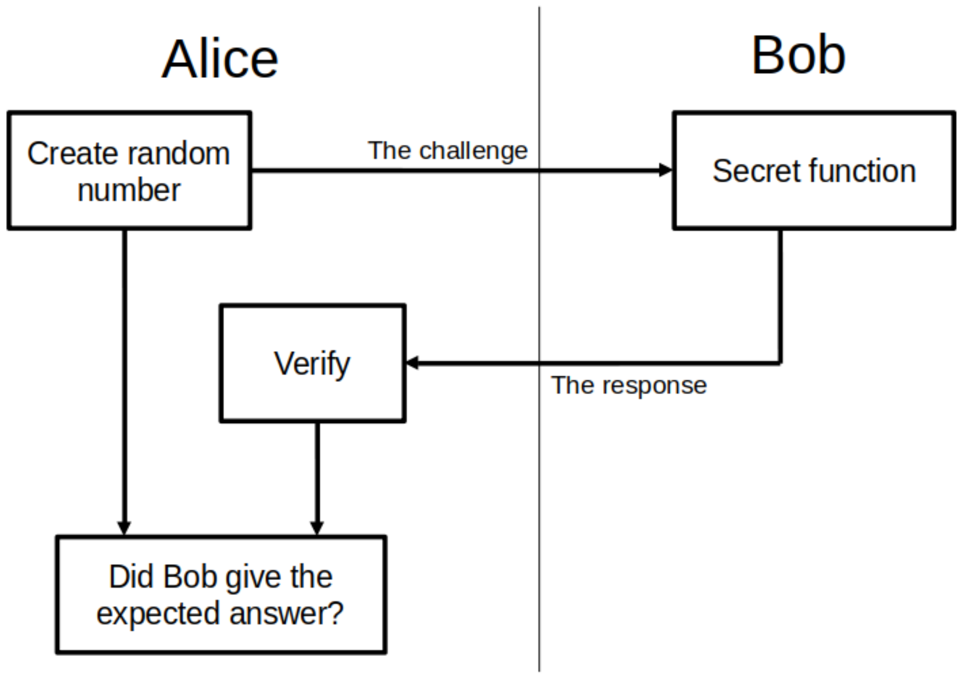


Figure 3.11: Block diagram of simple challenge-response algorithm

Physically unclonable functions (PUF) exploit manufacturing variations in order to create the secret function used by Bob. PUFs are 'black-box' functions whose output depends on the inputs. By creating them using manufacturing variations, the functions can be kept secret and are hard to duplicate. They are deterministic functions, that is, if the same input is given twice, the output will the the same on both runs. Some error correction is often needed to ensure this since the analog properties of the manufacturing variations do have some noise associated with them [4].

One possible PUF design is to intentionally introduce race conditions between logic gates, which is refereed to as an 'Arbiter PUF'. Each logic gate inherently has a delay between the input and output and this delay varies from one gate to the next. In a race condition between two gates, one will win and it will usually always be the same one since the variation in delays for each gate are due to manufacturing variations. However, since delay time is also influenced by external factors such as temperature, supply voltage and other environmental conditions some form of error correction is also needed for reliable operation.

An example of a simplified Arbiter PUF circuit is given in figure 3.12. We can apply a given input (or challenge) (marked as X) which will cause the signals to take a different path through the logic gates. Due to variations in the delay, the signal that 'wins' the race (gets to the latch first) will depend on the input. This process creates a 'secret' function that is unique to every chip that is manufactured.

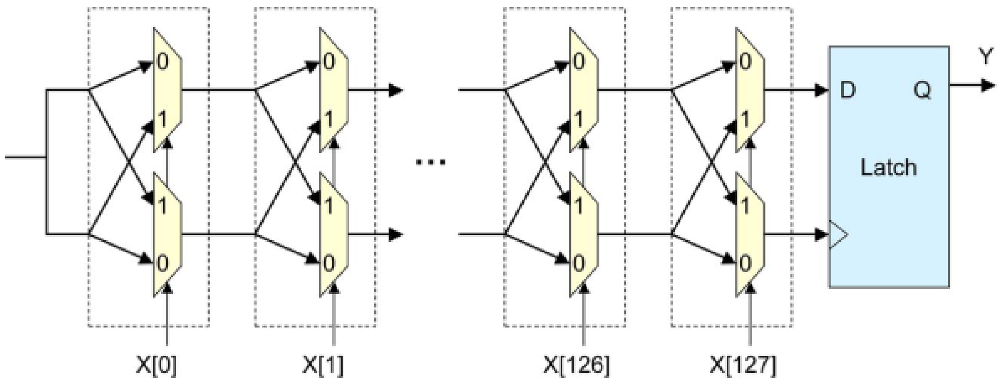


Figure 3.12: Arbiter PUF circuit [4] (© *2014, IEEE - reproduced with permission*)

One difficulty when using PUF for challenge-response algorithms is determining how Alice can verify the response is correct. Since the secret function appears random, is known only to the particular chip and ideally there is no way to extract/measure the secret function from the chip, Alice does not know what the function is. To deal with this Alice needs to build and store a table of valid challenge-response pairs for Bob's PUF chip. To do this she needs secure access to Bob's PUF chip where she applies random challenges, measures the responses and stores the results. Then this table can be used to verify the responses (as long as Alice issues challenges that are in her table). Since each challenge can only be used once, it is possible that Alice may use up all the challenge-response pairs in her table. Hence processes to renew the challenge-response pairs in the table may need to be taken into account during the design process. This table of challenge-response pairs is also a target for attackers and it becomes Alice's responsibly to keep it safe.

3.6.6 Static RAM power on state

Static RAM (SRAM) cells are built like the circuit diagram in figure 3.13 which is effectively two digital inverters in a feedback loop. This circuit can be in either of two stable states (A = low, B = high and vice-versa). Since the state will be held for as long as power is applied, this circuit can be used as volatile digital memory.

The state of the SRAM memory cell after power up and before programming (the 'power on state') is seemly random. Due the manufacturing variations between the transistors (variations in V_{th}) one of the gates will be slightly "stronger" than the other and tends to force the circuit into one of the states. This effectively creates a race condition between the transistors on power up. Since this race condition is mostly dependent on manufacturing variations, which don't change much over time, the power on state is likely to be the same every time the power is switched on. Hence this gives SRAM arrays PUF like characteristics (and PUF like security) but with no input data (output only) [4].

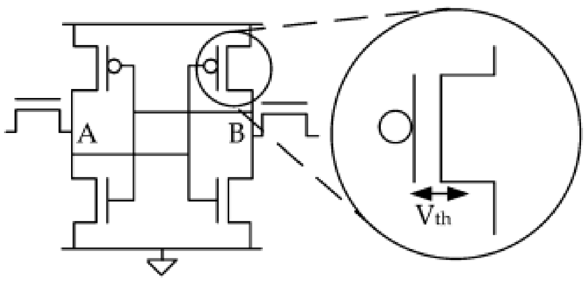


Figure 3.13: SRAM Circuit [4] (© *2014, IEEE - reproduced with permission*)

Whilst not useful for challenge-response due to the lack of input, it is possible to use SRAM to generate a random number which could then be used as a secret cryptographic key. Since the power on state is usually the same every time power is applied, the SRAM can also be used to store the random number (and it is non-volatile) in a way that makes it difficult to clone. Since the power on state may be influenced by environmental fluctuations and random noise, it is likely that some form of error correction is also required. The power on state can also be used for creating a digital fingerprint of the SRAM memory.

3.7 Conclusion

The four core objectives of information security, confidentiality, data integrity, authentication, and non-repudiation, can be achieved with a number of different technologies. While some of these have been developed and researched over hundreds of years, the demand for internet security has rapidly increased the research efforts in information security technology.

We have found that the technologies used to achieve the objectives fall into four distinct methods used: cryptography, widely witnessed, hardware security and exploiting manufacturing variances/defects.

Cryptography is the oldest and most well researched of these methods and is widely used to address all the aims of information security. Provided that the systems implementing the cryptography are designed and built correctly, and the secret key can be kept secret, cryptography can offer very strong protection against even the most determined and well-resourced attackers.

Hardware security is also in widespread use and is often used in conjunction with cryptography to improve the security of the system. This category includes a wide variety of devices where the electronic circuits and/or ICs are designed to store data and software in a way that is hard to access or modify. The security on these devices can usually be defeated by chip-off and IC de-packaging techniques however these approaches are very expensive and time consuming. This often limits its use to applications where potential attackers have limited resources to break the security.

The aim of data integrity can also be met by using widely witnessed systems where, after the data has been witnessed by a large number of people, the data can not be modified without detection. While not as widely used as cryptography or hardware security, this concept is becoming more popular due to recent developments in blockchain technology allowing the widely witnessed concept to work on commercial scale. If implemented correctly, the only way an attacker can break this system is to be in control of a majority of the witnesses (a 51% attack), an attack which can be made infeasible by simply having a large enough number of witnesses.

Exploiting manufacturing variances/defects is a relatively new area that is still mostly in the research phase with limited commercial applications. However it appears to offer some innovative solutions to the data integrity and authentication aims. These are aims that cryptography has often struggled with in applications where public keys can not be easily trusted. While a number of different applications in this area are been explored, PRNU and the 'Digital Fingerprint' concept have had the most success in digital forensics.

Overall, the best security is achieved when a number of these methods are used together. Cryptography alone is only as secure as the key management/storage systems. Cryptography and hardware security combined has offered improved security in many applications. The recent addition of widely witnessed processes to many systems has improved security even further.

3.7.1 Relevance to the Digital Evidence Bag

Going back to the four requirements of the digital evidence bag (from section 2.5.1), tamper evident, unforgable, clean and offline, it is apparent that the tamper evident and unforgability requirements are very closely aligned to the data integrity and authentication objectives of information security. Hence we expect that the technological solutions discussed above would be applicable to a DEB.

Widely witnessed systems may offer good solutions for the tamper evident requirement, however they conflict with the offline requirement since all widely witness systems rely on distribution via the internet to the many witnesses. As such, the rest of this thesis focuses on applying manufacturing variances/defects and to a lesser extent, hardware security solutions to the design of a DEB.

Chapter 4

Digital Memory Fingerprinting

4.1 Introduction

The aim of this section is to create a method for digital fingerprinting (as defined in section 3.6.2) off-the-shelf digital memory chips. We propose that there are variances in the manufacturing of the transistors within the memory chips that will affect the output voltages of the individual memory cells. CMOS logic defines a binary '0' output to be between 0V and 0.45V and a binary '1' output to be between 2.4V and 5V [5]. This is a very large range of accepted values and from the manufacturer's point of view, as long as the output is somewhere in that range, the memory chip meets the specifications.

So we propose using an Analog to Digital Converter (ADC) to measure the output voltage of the memory chips. This would be carried out by sequentially reading from every memory address and performing an ADC conversion for each bit of each address read out. The data stored in the memory would need to be the same each time the fingerprint is read. Either clearing the memory to all '0's or setting it it all '1's before reading the fingerprint would achieve the best result. However, partial fingerprint matches may be achievable with only some of the bits in the required state.

Then we expect that two fingerprints read independently from the same memory chip would have a high correlation. We also expect that two fingerprints read from different memory chips would have a low correlation. This can then be used to identify individual memory chips and detect forged memory chips.

In relation to the digital evidence bag from section 2.5.1, the DEB must contain a memory element to store the data that makes up the digital evidence. By using a digital fingerprint we expect to achieve the unforgeable requirement by exploiting the manufacturing variances within the memory component of the DEB. This is the one requirement that has not been met by the existing best-practice solution of using write-once memory [18] and a digital fingerprint may provide a simple method to ensure unforgeability.

4.1.1 Background

From section 3.6.2, we know that a digital fingerprint needs to be created by measuring manufacturing variances and defects. It also needs to be unique to each memory chip and the measurements need to be repeatable (within an acceptable margin of error).

Digital camera fingerprints

The concept of a digital fingerprint has been deployed in the context of Photo Response Non-Uniformity (PRNU), a process used to uniquely identify individual digital cameras [38]. The image sensor within a digital camera is made up of millions of individual photo-diodes (one for each pixel of the image), each of which converts the light intensity to an analog voltage. This voltage is then fed through an ADC and further processing to create the digital image. Due to manufacturing variances each photo-diode reacts slightly differently to a given light intensity and some simply don't work at all (called "dead pixels"). PRNU exploits this variation between the photo-diodes to create a digital fingerprint for the camera.

One of the issues found in PRNU was that the measured fingerprint comes from more than just the image sensor [41]. Each camera lens also has manufacturing variances (and dust that has fallen on lens surface) that contribute to the fingerprint. In addition to this, there are other electronic components such as amplifiers, buffers and ADCs within the camera that may also contribute to the fingerprint. Hence when designing experiments to measure the fingerprint, we need to take into account all components of the system.

Memory fingerprinting

Some work has been reported, investigating a digital fingerprint for NAND flash memory devices. Measuring the time taken to write to each cell has been shown to provide a digital fingerprint for the device [42] [39]. There are manufacturing variances between each floating-gate transistor in the memory array which affect how long it takes for enough charge to build up on the floating-gate to set a given cell to a binary '0'. To measure the write time of each memory cell, it is necessary to clear the memory and then write to every cell, a process which is destructive to any data that is stored in the memory. Memory cells are usually rated for a given number of 'write-cycles' and after that many writes have taken place, the ability for the cells to reliably store data is no longer guaranteed. Hence measuring the write time to create the fingerprint can shorten the life of the memory device, particularly in applications where the fingerprint needs to be measured regularly.

When NAND flash memory chips are manufactured, it is expected that a small number of memory cells simply won't work at all due to manufacturing defects. To deal with this problem, memory control systems are designed to identify and isolate failed cells so that the user's data is only stored on the working cells. However we can exploit the failed cells to create a digital fingerprint for the memory chip. The number and location of the failed cells is unique for each chip, so by measuring the bit-errors, we can create a map of where the failed cells are [40]. It was also found that bit-errors are caused by other factors such as storage temperature and the number of write cycles that the cells had been through. Since these factors change over time, any testing to match a digital fingerprint needs to allow for these bit-errors that are not caused by manufacturing defects.

Using analog output voltages may offer a better method of identifying manufacturing defects to measure for the creation of a digital fingerprint. Since the analog output voltages can be measured with just a read operation, verifying a fingerprint may not be destructive to the data in memory, nor will it degrade the memory life-span like extra write-cycles would.

4.2 Experiment overview

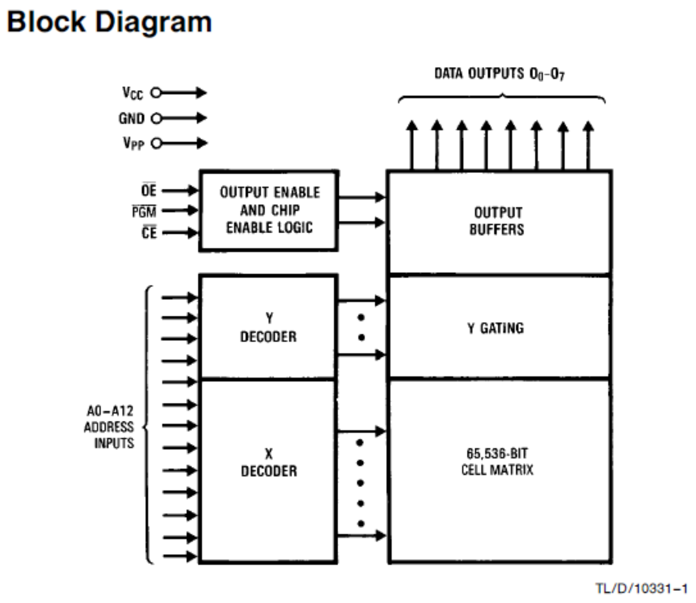
We designed an experiment to test if a digital fingerprint of an Erasable Programmable Read-only Memory (EPROM) chip could be created by exploiting the variations in analog output voltages. To do this we propose to measure the output voltage (using an ADC) of every memory cell from 3 different chips. These measurements would then be repeated 3 times. We expect to observe variations in the output voltage between cells. We also expect that if the output voltage of the same memory cell is measured twice, then both measurements should be approximately the same (allowing for measurement uncertainty and background noise).

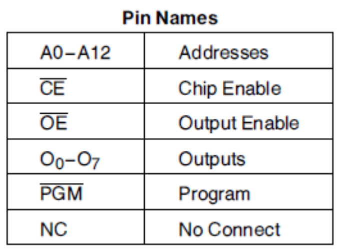
After reading every cell in the memory chip, a dataset is created which is an array of voltage measurements. Correlation is used to compare the datasets from each read of the memory chips. To use a digital fingerprint for identify individual memory chips, we require the correlation between datasets from the same chip to be consistently larger than the correlation between datasets from different chips.

4.2.1 Memory Architecture

For this experiment we used EPROM memory chips since they offered a relatively simple interface. Two models were tested, TMS2764 which is a NMOS device and the 27C64 which is a CMOS device. Both of these devices contain 65536 bits of digital memory which is accessed via a 8 bit parallel data bus and 13 bit address bus. The diagram in figure 4.1 shows the internal structure of these memory chips.

Within the memory chip there are digital output buffers between the memory cells and the output pins. At the very least, we expect the output buffers to contribute to the digital





Pin Names

A0–A12	Addresses
$\overline{CE}$	Chip Enable
$\overline{OE}$	Output Enable
O_0–O_7	Outputs
$\overline{PGM}$	Program
NC	No Connect

Figure 4.1: Internal block diagram of EPROM memory chips [5] (*Courtesy of Texas Instruments Inc.*)

fingerprint. The buffers may even effectively block any analog effects from the memory cells from reaching the output.

4.2.2 NMOS vs CMOS

While there are a number of logic gate architectures using transistors (and resistors), the most common one is Complementary Metal–Oxide–Semiconductor (CMOS) where a N-type transistor is used to pull the gate output low and a P-type transistor is used to pull the gate output high. We also considered at N-type Metal-Oxide-Semiconductor (NMOS) logic gates where a pull-up resistor is used to pull the output high. To output a low signal, a N-type transistor is switched on to drive the output low. Figure 4.2 shows the differences between NMOS and CMOS using an inverter gate as an example.

CMOS logic is much more common, in part due to its reduced power consumption. When the output is low, there is negligible current draw for the CMOS gate, where for the NMOS gate current will flow through the resistor and switched on transistor. Both cases have no current draw when the output is high. Although not immediately apparent, the current through the transistor(s) is relevant to how the output voltage is measured.

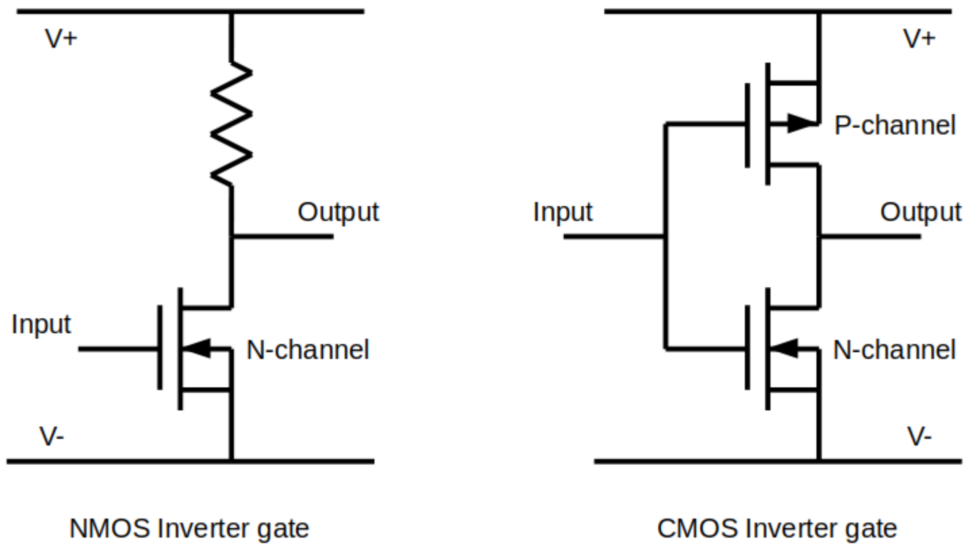


Figure 4.2: Circuit diagrams of NMOS and CMOS inverters

Effects of the transistor currents

When the MOSFETs in the logic gates are switched on, the current-voltage relation between the drain and source pins is roughly similar to a resistor. That is, the voltage drop across the device increases as the current increases (until saturation is reached). It is this current-voltage relation, or "effective resistance", of the MOSFETs that we expect to vary between transistors within the memory chip. To build the fingerprint, we can measure the effective resistance of each MOSFET by measuring the voltage drop across the MOSFET. In this experiment, the ADC connected to the output draws a negligible current. This causes two different effects to be observed in NMOS vs CMOS devices.

In the NMOS case, current only flows when the output is in a low state. In this state the output voltage is effectively determined by a voltage divider with the resistor and MOSFET. The voltage drop across the resistor is much greater than the voltage drop across the MOSFET causing the output to be low enough for a binary '0'. However the voltage drop across the MOSFET is still significant enough to be measurable by the ADC.

When the output is in the high state, almost no current flows and hence there is a minute voltage drop across the resistor. This suggests that the output voltage will be equal to the supply voltage and no information about the MOSFET is able to be measured (the supply voltage is the same for all gates on the same memory chip). This is not useful for creating a fingerprint so we need to ensure the outputs are all in the binary low state before measuring the output voltage.

In the CMOS case, there is no significant current flow in both the high and low states. This causes a problem since the output will be equal to the supply voltage for all cells set to binary '1' and equal to zero for all cells set to binary '0'. This means we are unable to measure individual components to create a digital fingerprint. To deal with this problem, load resistors were attached to the output.

Eight load resistors were used, one connected between each data line and ground. Hence when the output is a binary '1' current will now flow through the P-channel MOSFET, the output and then the load resistor to ground. Similar to the NMOS case, this then causes the output voltage to represent the effective resistance of the P-channel MOSFET, which can be used to create the digital fingerprint.

4.3 Experimental setup

The measurements were made using a PIC4553 microcontroller [43] which contains a 13 channel, 12 bit successive approximation ADC, giving a resolution of $1.2mV$ with a nominal $5V$ reference voltage. The data outputs from the memory chip were connected directly to the inputs of the ADC as shown in figure 4.3. The PIC microcontroller was also used to supply the necessary signals to the address and control lines to facilitate a read operation. The ADC output data was then transferred to a computer via the PIC microcontroller's built in USB interface.

4.3.1 Impact of the test circuit

We expect elements of the test circuit such as the ADC and load resistors to contribute to the digital fingerprint that is measured. In order to differentiate these elements from those within the memory chips, care was taken to use the same test circuit configuration for all measurements that were taken. It is important to note that the same 8 resistors were used as the load resistors for all measurements. By doing this, any correlation between data from different memory chips can be attributed to elements of the test circuit or the measurement process used.

4.3.2 Memory read method

To minimize the impact of any random noise, each address is read 32 times, with the average of the 32 reads then been used. We have intentionally added a $1ms$ delay between switching and starting the ADC read process to allow any transients to settle after the address bus and read enable bit have been set. Reading each address involves 8 ADC conversions, one per bit. Between each bit there is a delay of $1\mu s$.

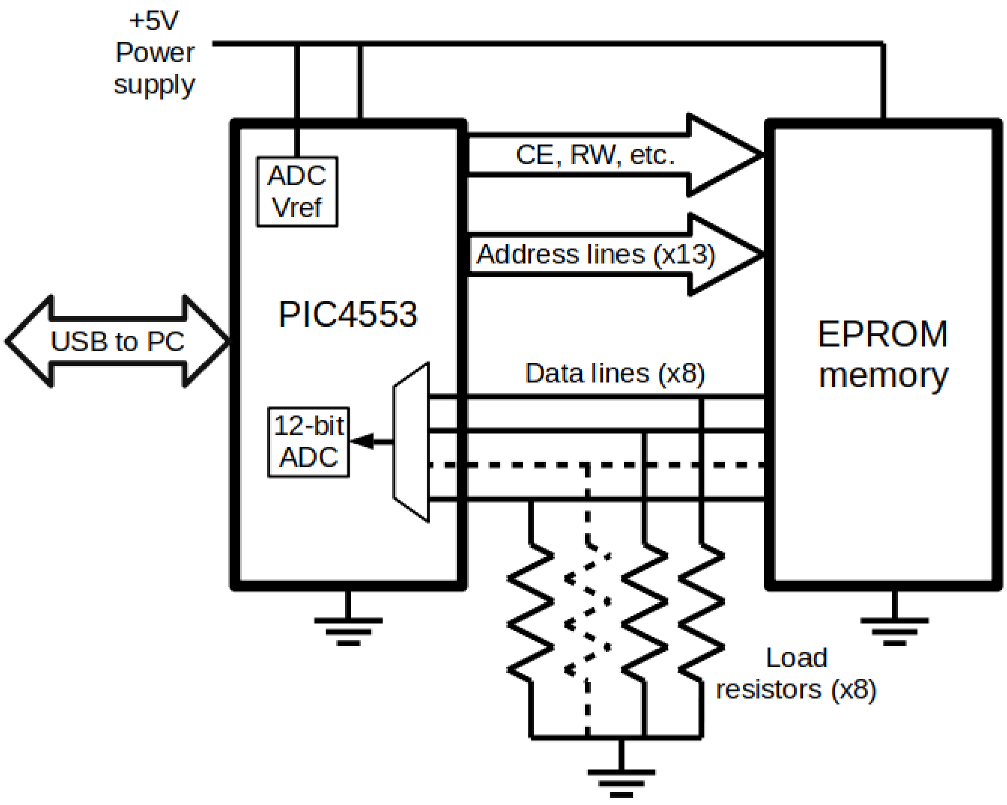


Figure 4.3: Circuit used to measure the analog output voltages

Three memory chips of each type were tested, each chip being read three times. Hence nine datasets were produced for each of the two chip types. Between each read operation the system was powered down and allowed to cool while the memory chips were swapped. The order of the read operations was as follows:

1. Memory chip 1, read 1
2. Memory chip 2, read 1
3. Memory chip 3, read 1
4. Memory chip 1, read 2
5. Memory chip 2, read 2
6. Memory chip 3, read 2
7. Memory chip 1, read 3

8. Memory chip 2, read 3

9. Memory chip 3, read 3

The process was first used on the NMOS memory chips before being repeated with the CMOS memory chips.

4.3.3 Power supply

There were three items in the memory read circuit that required a nominal 5V power supply: the PIC4553 microcontroller, Vref for the ADC and the EPROM memory chip. Three different power supply options for these were explored:

- 5V power bus from the computer's USB port
- Four 1.5V alkaline batteries (in series with two power diodes)
- A laboratory variable voltage, 3A DC switch-mode power supply

We found the USB power supply produced uneven and unrepeatable results as shown in figure 4.4. We expect this is due to other peripherals attached to the computer and other tasks the computer is working on during the reading process.

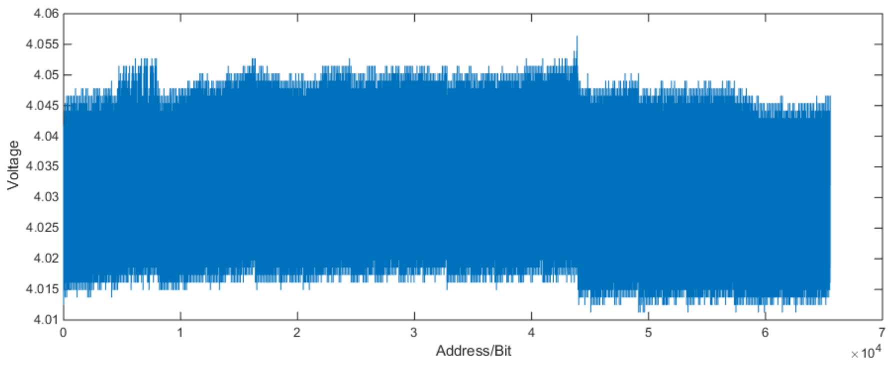


Figure 4.4: Memory cell output voltage when the memory chip is powered by the USB bus

The battery power supply produced a very smooth and noise free result as shown in figure 4.5. It can also be seen that the average output voltage slowly drops over time during the reading of the memory chip. We suspect this is due to the batteries slowing discharging and hence the terminal voltage reducing. However the ADC Vref and memory chip power are both supplied from the same batteries which should cancel out any gradual voltage changes.

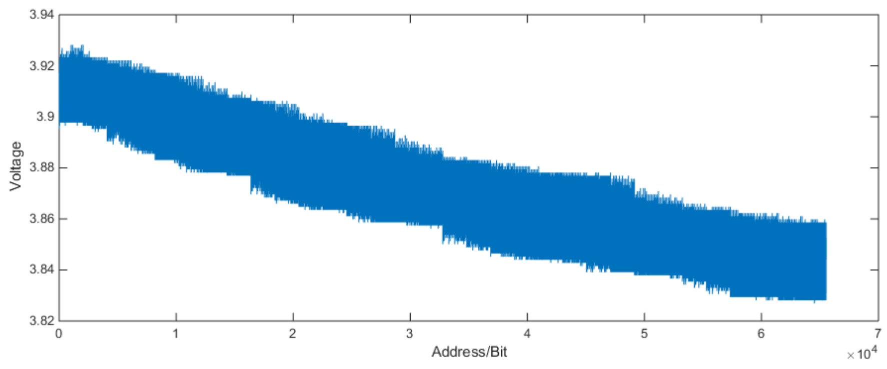


Figure 4.5: Memory cell output voltage when the memory chip is powered by alkaline batteries

As shown in figure 4.6, the laboratory switch-mode power supply produced the most stationery result, and the result with the least visible noise, so the switch-mode power supply was used for all subsequent tests.

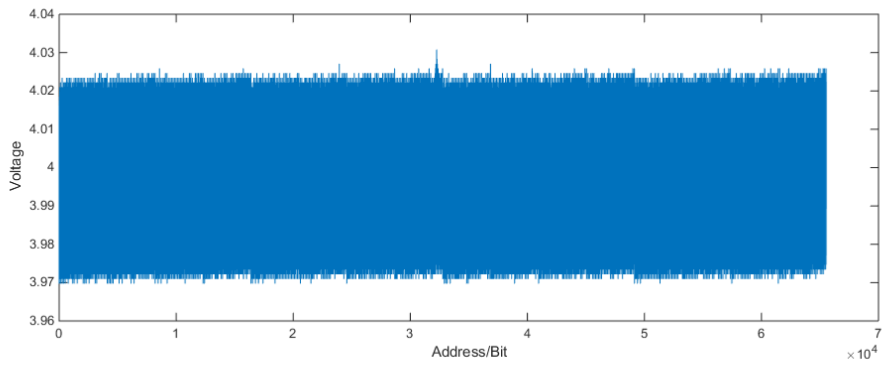


Figure 4.6: Memory cell output voltage when the memory chip is powered by the switch-mode power supply

4.3.4 Temperature Effects

Reading the entire memory chip took over 10 minutes to complete. It was quickly noticed that readings near the start were slightly lower as shown in figure 4.7. We suspect this is due to the system being cold while starting and then slowly warming up to a steady state. This is supported by the fact that these variations were not present when reading the entire memory again immediately after the first read had finished.

To minimize the effect of temperature on our results, each memory chip was read entirely

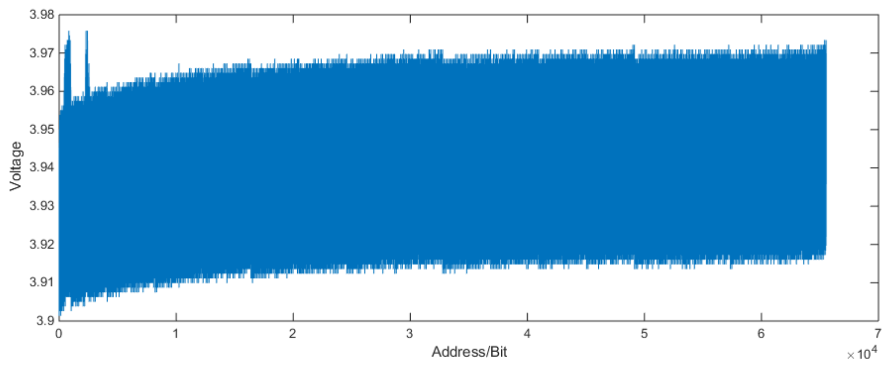


Figure 4.7: Memory cell output voltage when the memory chip is cold when the read process starts

four times in immediate succession. An example of four successive read operations can be seen in figure 4.8 where the cold start is visible on the 1st read. We then selected one of the read data sets that was least affected by temperature for further calculations.

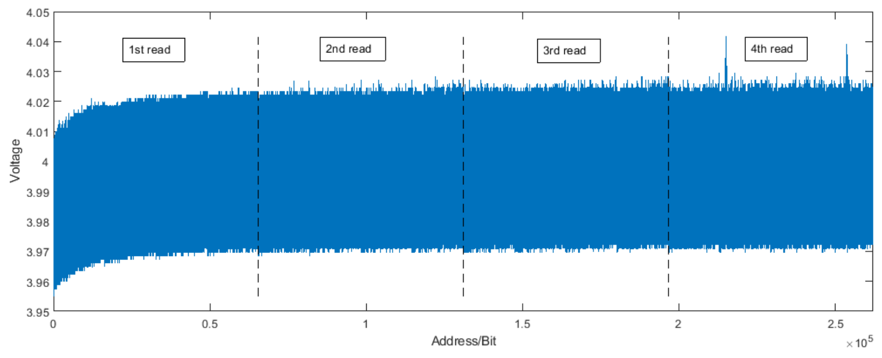


Figure 4.8: Four consecutive reads from one memory chip, showing the later reads are not affected by the cold start

4.4 Results

4.4.1 NMOS (TMS2764)

This section explains how the 9 datasets from the NMOS memory chips were analyzed.

Correlation results

Each dataset was first normalized to have an average of zero and standard deviation of one. The results of correlating each dataset with each other one is shown in Table 4.1. It can be seen that there is a very strong correlation between datasets from the same chip ($> 99.8\%$). However it is also worth noting that the correlation between different memory chips is higher than expected ($95\% - 99\%$). This suggests that some other effect not related to the manufacturing variances of the memory chips is causing the high correlation results, or that the manufacturing variances are non-existent or negligible in size.

		Read 1			Read 2			Read 3		
		Chip 1	Chip 2	Chip 3	Chip 1	Chip 2	Chip 3	Chip 1	Chip 2	Chip 3
Read 3	Chip 3	98.5%	97.7%	99.9%	98.5%	97.4%	99.8%	98.5%	97.4%	100%
	Chip 2	95.8%	99.9%	97.4%	95.9%	99.9%	97.3%	96.1%	100%	
	Chip 1	99.9%	96.5%	98.6%	99.9%	96.1%	98.5%	100%		
Read 2	Chip 3	98.4%	97.6%	99.9%	98.5%	97.3%	100%			
	Chip 2	95.8%	99.8%	97.4%	95.9%	100%				
	Chip 1	99.9%	96.4%	98.5%	100%					
Read 1	Chip 3	98.5%	97.7%	100%						
	Chip 2	96.3%	100%							
	Chip 1	100%								

Table 4.1: Correlating each result set from the NMOS memory chips (shaded cells represent correlation between data from the same memory chip)

Separating each data bit line

Upon looking closely at the data it became apparent there was a pattern that repeated every eight values which corresponded to each bit on the data bus. So each dataset was split into eight separate datasets, each corresponding to the data from one of the eight bit lines. This can be seen in figure 4.9. There is a noticeable difference in the average output voltage for each bit line. And by comparing with figures 4.10 and 4.11, it can be seen that the average output voltage for each bit line varies between memory chips. This supports the theory that each output buffer of the memory chip is unique and that the output voltage varies slightly between all output buffers due to variations in the manufacturing process.

The data for each bit line was normalized and correlated to the data from the other read operations that corresponded with the same bit line. For each dataset correlation, there were 8 results, one per bit line. Table 4.2 shows the average of each set of 8 correlation results. This allows us to exclude any impact from the output buffers. It can be seen that all the correlation values are all still reasonably high and there is no noticeable difference

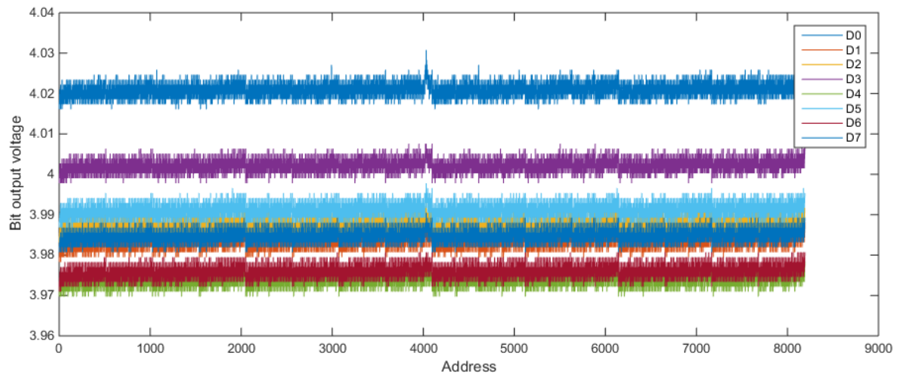


Figure 4.9: Data from chip 1, read 1 separated by each bit line in the data bus (NMOS)

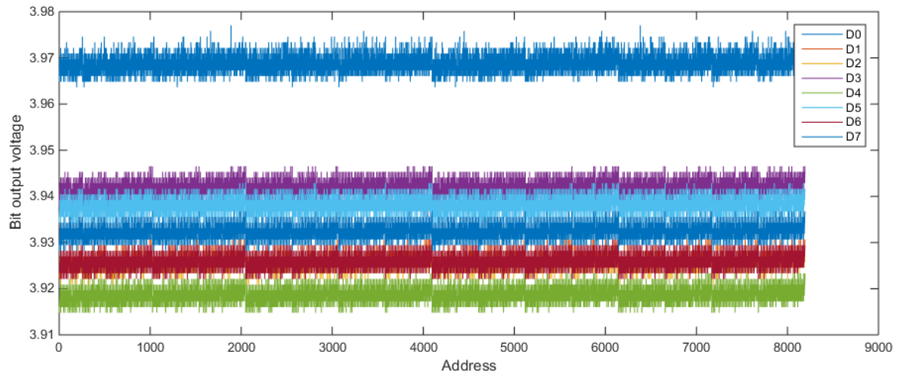


Figure 4.10: Data from chip 2, read 1 separated by each bit line in the data bus (NMOS)

between data from the same chip and data from different chips.

Address Bus Effects

By looking closely at the data from the ADC, there appears to be a link between the state of the address bus and the corresponding output voltage of the data lines. We noticed that the output voltage was larger when the address bus had more bits set to '1' as shown in figure 4.12. This effect was only seen in the NMOS chips and not the CMOS chips.

We believe this is due to parasitic resistances in the power supply circuit. Due to the way it is built, NMOS technology draws more current when gates are in the '0' state than in the '1' state. As a result, when the address bus has more bits in the '0' state the memory chip as a whole draws more current from the power supply. And due to parasitic resistances in the power supply wires, this results in the power supply voltage dropping slightly which would also cause the output voltages to drop.

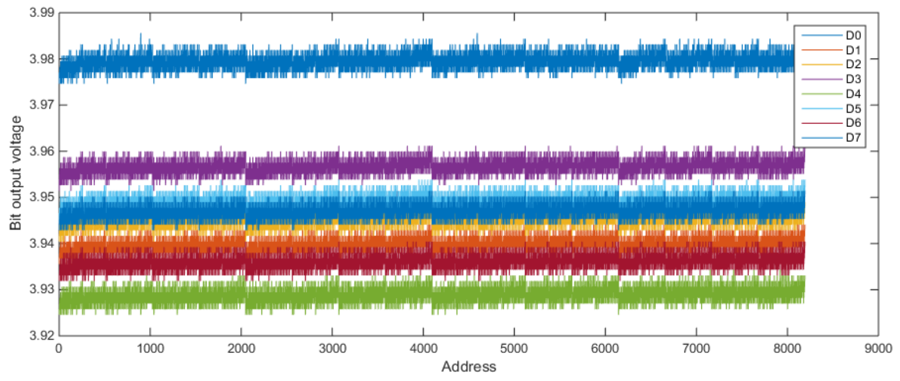


Figure 4.11: Data from chip 3, read 1 separated by each bit line in the data bus (NMOS)

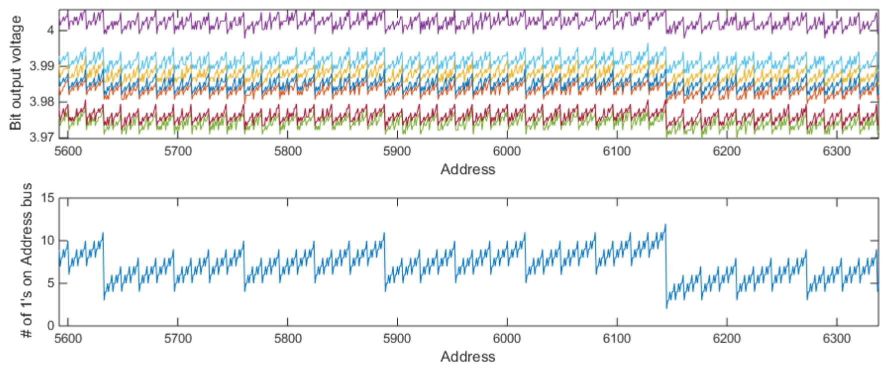


Figure 4.12: A close up of the data from NMOS chip 1, read 1 showing the effect of the address bus state

NMOS Results Conclusion

The high correlations due to the address bus effects are essentially another part of the circuit interfering with our ability to measure the manufacturing variations. The variations between cells due to the address bus effects don't appear to contribute anything useful to the digital fingerprint since the variations are the same for every chip. More importantly, there is no noticeable difference between the correlations for data from the same chip and data from different chips. This suggests that every factor (excluding the buffers) that contributes to the output voltage is part of the measurement circuit or is consistent across every memory chip produced. Which leads to the conclusion that variation between the output voltage of individual memory cells is either non-existent or, more likely, is masked by the output buffers.

		Read 1			Read 2			Read 3		
		Chip 1	Chip 2	Chip 3	Chip 1	Chip 2	Chip 3	Chip 1	Chip 2	Chip 3
Read 3	Chip 3	90.5%	90.5%	90.1%	89.7%	89.1%	88.6%	90.3%	90.7%	100%
	Chip 2	91.1%	90.9%	90.6%	90.6%	89.4%	89.4%	90.9%	100%	
	Chip 1	91.0%	90.8%	90.9%	89.7%	88.4%	88.9%	100%		
Read 2	Chip 3	88.6%	88.8%	89.0%	88.2%	86.6%	100%			
	Chip 2	89.7%	89.2%	87.7%	88.7%	100%				
	Chip 1	90.0%	90.1%	90.2%	100%					
Read 1	Chip 3	91.0%	90.4%	100%						
	Chip 2	91.0%	100%							
	Chip 1	100%								

Table 4.2: Correlating the NMOS data from each bit line separately (shaded cells represent correlation between data from the same memory chip). Each value shown is the average of the 8 correlation results for each bit line. The full results are shown in Appendix A.

4.4.2 CMOS (27C64)

Initially, we found that the output from the CMOS memory chips showed almost no variation at all. The output voltage was always equal to the supply voltage as shown in figure 4.13. Hence 8 load resistors was added between each data output and the $0V$ rail as discussed in section 4.2.2. The resistors used were 1% tolerance resistors with measured values between $1.45k\Omega$ and $1.46k\Omega$. Care was taken to ensure the same resistors were connected in the same order for all the measurements taken. Like with the NMOS chips, three different CMOS memory chips were tested, each being read three times over.

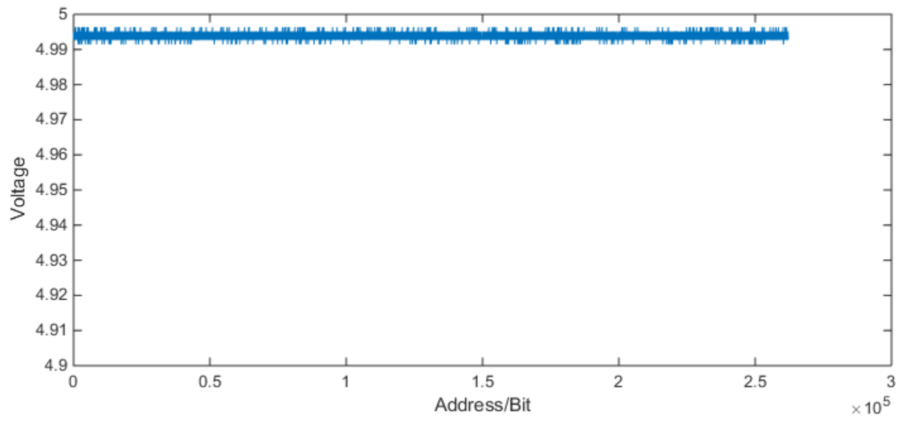


Figure 4.13: Data from CMOS memory chip with no load on the output

Correlation result

The results from correlating the output data from each memory chip is shown in table 4.3. The process used here was the same as with the NMOS chips and the results are very similar. There is a very strong correlation between data from the same chip and a reasonably high correlation between data from different chips.

		Read 1			Read 2			Read 3		
		Chip 1	Chip 2	Chip 3	Chip 1	Chip 2	Chip 3	Chip 1	Chip 2	Chip 3
Read 3	Chip 3	96.3%	96.3%	99.9%	96.7%	97.6%	99.9%	96.8%	97.8%	100%
	Chip 2	95.1%	99.9%	97.7%	95.6%	99.9%	97.4%	95.7%	100%	
	Chip 1	99.9%	95.7%	97.1%	99.9%	95.9%	96.6%	100%		
Read 2	Chip 3	96.1%	97.0%	99.9%	96.5%	97.1%	100%			
	Chip 2	95.2%	99.9%	97.5%	95.7%	100%				
	Chip 1	99.9%	95.6%	97.0%	100%					
Read 1	Chip 3	96.6%	97.4%	100%						
	Chip 2	95.1%	100%							
	Chip 1	100%								

Table 4.3: Correlating each result set from the CMOS memory chips (shaded cells represent correlation between data from the same memory chip)

Separating each data bit line

Like with the NMOS devices, we then separated the data sets into individual bit lines, which revealed that there was almost no variation at all in the output voltage between memory cells (address locations) of the same output bit. Figure 4.14 shows the comparison between the bit line data for the three chips where it can be seen that the memory address being read from has no effect on the output voltage. The small notches on bits are due to quantization errors in the ADC where the voltage is somewhere between two digital values (the quantization interval is $1.2mV$).

CMOS Results Conclusion

There are two conclusions that can be drawn from figure 4.14, the first being that the address bus state has no effect on the output voltage which is to be expected from a CMOS device where device current draw is not dependent on the state of the logic gates. And similar to the NMOS devices, we can also see that any variation between individual memory cells is either non-existent or, most likely, is masked by the output buffers.

The data from each of the individual bit lines shows almost no variance, meaning the variance is less than the quantization level. Hence correlating these data sets doesn't offer any meaningful results.

4.5 Conclusion and future work

4.5.1 Output buffer fingerprinting

Whilst we were unable to measure any analog effects from individual memory cells, we have shown that the output buffers can produce a digital fingerprint-like effect. In both the NMOS and CMOS chips, this could be observed as the slightly higher correlations between data sets from the same memory chip.

This offers some ability to identify memory chips since the output buffer fingerprint is different between different memory chips. However, due to there being only 8 output buffers, the output buffer fingerprint is unlikely to be unique within a large number of memory chips. The reason for this is that an array of 8 scalar measurements is much less likely to be unique than an array of 65536 scalar measurements. The lack of uniqueness limits its usefulness as a digital fingerprint since a different chip with the same (or similar) output buffer fingerprint can feasibly be found by simply testing numerous memory chips until one that matches is found.

Referring back to the digital fingerprint definition in section 3.6.2, the uniqueness requirement is not met when creating a fingerprint from just the 8 output buffers.

4.5.2 Buffer Problem

Although we were unable to measure any variations in output voltages between individual memory cells, it is still likely that the variations are there. The memory cells are built with transistors in a similar way to the other logic gates within the chip. We have shown that the logic gates that makeup the output buffers have a measurable and repeatable variation in output voltage so we expect the memory cells will as well.

In order to measure and verify this theory, we need to test on a circuit without digital logic gates or buffers between the memory cells and the ADC. Since every commercially available memory chip contains some digital logic between the memory cells and the output, it would be necessary to design and test on a custom built memory chip that allows the individual memory cells to be measured directly by an ADC. Due to the lack of availability of discrete floating gate MOSFETs, and the complexity of even a modestly sized memory, building a custom memory chip would require fabricating a custom integrated circuit.

4.5.3 Flash Memory

EPROM memory is rarely used now and flash memory dominates in the market for floating gate MOSFET based memory devices. While the transistors and logic gates are built the same in both, there are differences in the control circuits.

Flash memory devices are usually built with a controller placed between the interface and the memory cells. The controller performs a number of functions including wear-leveling, detecting and isolating failed blocks and mapping addresses to physical blocks of memory. Each input address (or logical address) is mapped to a different part of the physical memory array, and when choosing this mapping, the controller will avoid using failed blocks. Since memory cells wear out after a certain number of uses, the controller also aims to map data onto physical blocks that have had the least amount of use so as to make sure all the memory cell wear out evenly.

Due to the controller, it can be difficult to know which physical cells are accessed for a given logical address. When creating a digital fingerprint, this causes issues since it is necessary to know which physical elements are being measured. Hence for this process to work on flash memory, direct access to the physical memory cells is required, which is often not available in off-the-shelf flash memory devices. However with a custom built memory chip, it should be possible to create a digital fingerprint for flash memory devices.

4.5.4 Relevance to the digital evidence bag

In the context of the DEB, the aim of the digital fingerprint is to meet the unforgeability requirement. While measuring the output voltage variations of the buffers has failed to meet the unforgeability requirement, this method does offer some small advantages. It does offer the ability to detect incidental errors, a process that is currently done using paper audit logs.

Since the DEB is likely to be a custom built device, the use of a custom built memory chip within it is a feasible possibility. With a custom built chip it would be possible to have the ADC connected directly to the memory cells. This would allow the voltage variations of the memory cells instead of the output buffers to be measured. Given that there would be millions of memory cells, a digital fingerprint derived from them would most likely be unique and hence the unforgeability requirement would be met.

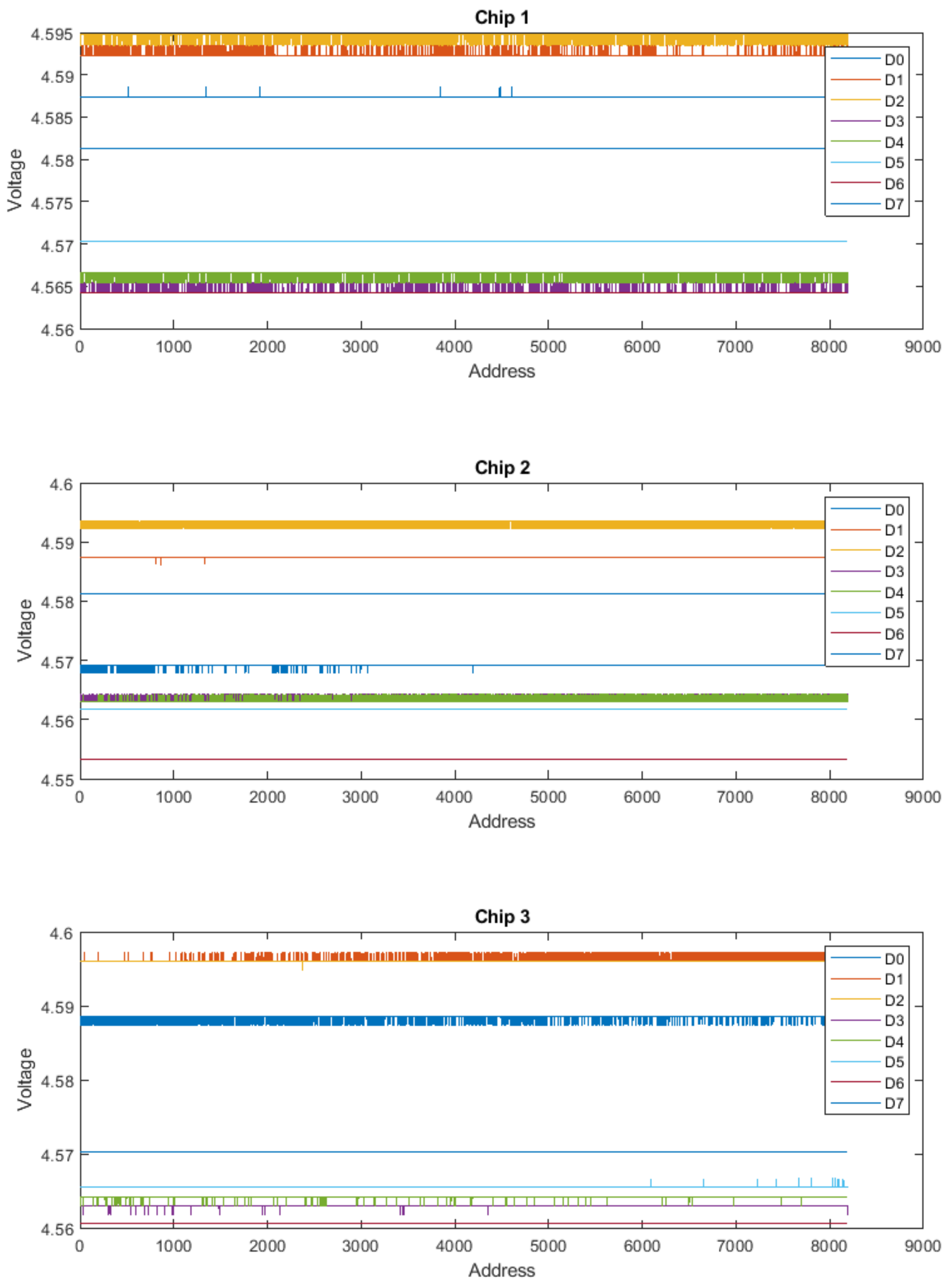


Figure 4.14: Data from each of the three CMOS chips split into individual bit lines

Chapter 5

An Analog Memory DEB

5.1 Introduction

The ‘perfect’ nature of digital data can be problematic for forensic investigations which traditionally rely on observing small imperfections to determine an object’s history. For example, film photography was not only difficult to forge but it was also easy to detect copies, forgeries and alterations by examining the film for small imperfections. Inferring details about, or even proving, the source of a film image was also easier to accomplish.

While small imperfections still exist in digital electronics, they are filtered out and corrected at every stage of the circuit by design. This allows digital devices to have better reliability and suffer no loss of data quality. In this chapter we look at storing digital data in analog memory cells in order to exploit some of the imperfections and deliberately cause a loss of quality in order to provide forensic information for authentication and data integrity purposes. We then use this to propose a design for a digital evidence bag (DEB) based on analog memory.

5.1.1 Basic model

An analog memory cell is a memory cell that can store a value that is anywhere within an expected range of values. Unlike digital memory cells which can only be in one of two states, analog memory cells have a continuous scale of states (and hence infinite) they can be in. Due to this, it is impossible to read the exact same value out of memory cell as when it was written to the memory. In practical implementations, they are also affected by random noise and degradation over time.

In this chapter we work with a model of an abstract, idealized analog memory cell without considering how this would be built in practice. However some discussion around practical circuit theory is necessary to fully understand how analog memory cells can operate.

In an electronics sense, we can think of these as devices that store a given charge for a period of time. Once a voltage has been written to the cell it should hold this value with some degree of accuracy. The "water-analogy" commonly used in electronics theory for this case is a bucket of water as shown in figure 5.1. Writing to the memory cell is adding (or removing) water to the bucket until the water is at the desired level. The water level in the bucket represents the value stored in memory. The water level can be read back at any later time and it will be roughly the same as when it was written.

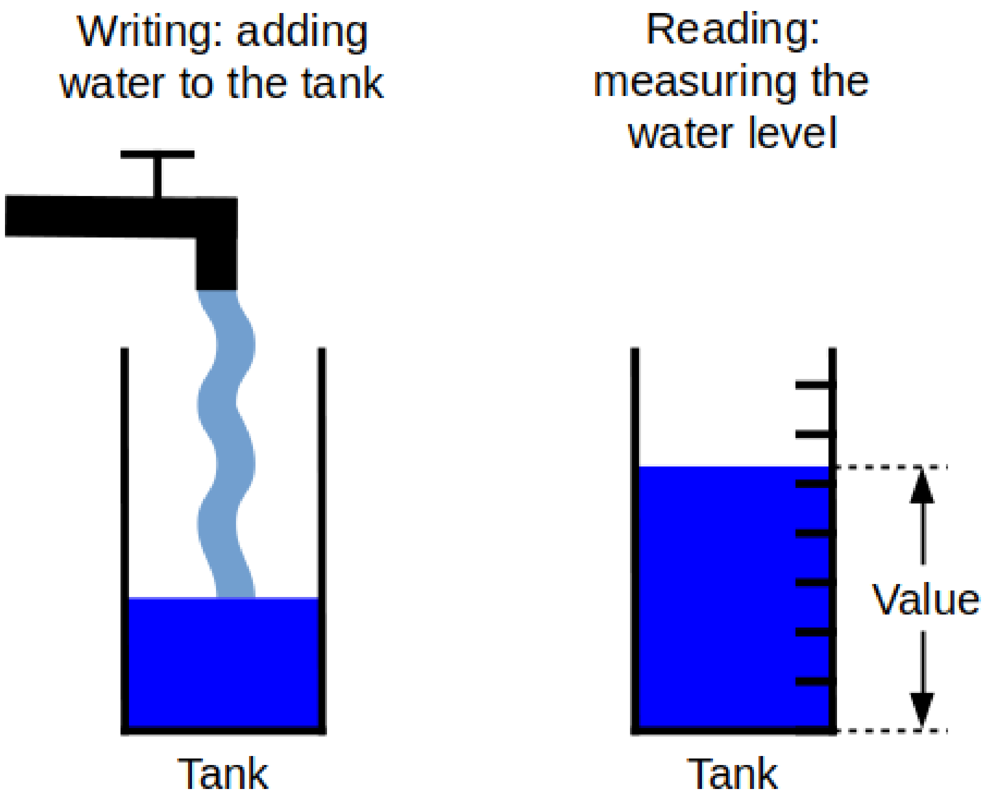

Writing: adding water to the tank
Reading: measuring the water level
Value
Tank
Tank

Figure 5.1: Water analogy for a analog memory cell

5.1.2 Characteristics

Like with digital memory, analog memory can be either volatile or non-volatile. However it is not always clear if a memory cell is volatile or non-volatile. Over time the value stored in the cell will degrade and drift away from the value that was originally written. The time spans for this range from seconds to several years depending on the technology used and the amount of variation from the original value that can be tolerated. Active control circuits can be built around the cell to slow and/or correct any drift in the cell however this control circuit then makes the overall device a volatile memory device.

Writing

The concept of writing to an analog memory cell suggests that the write process sets the value in the cell to be equal to some value of our choosing that we feed into the memory control circuit. Any previous value stored in the memory cell is irrelevant and will be erased/written-over when the write process is initiated.

Many memory technologies use an erase operation before writing to set all the cells back to zero (or some other set value). However if the erase operation is removed, the write operation will just add the new value on top of the existing value in the memory cell. In the digital realm, this doesn't mean much since the memory cells are only in one of two states but in the analog realm, it allows us to create what we call "additive analog memory". With this, the new value to be written is added (or subtracted in the case of writing negative values) to the existing state of the cell. As with all analog electronics, there will also be a level of noise.

$$M_a(i) = M_b(i) + V(i) + n$$

$M_a(i)$: state of memory cell i after the write operation
$M_b(i)$: state of memory cell i before the write operation
$V(i)$: value to be written to memory cell i (can be negative)
n: noise

In practice, there are also limits to the minimum and maximum values that can be stored in the memory cell. Any attempt to do a write operation that exceeds these limits will usually just leave the value in the memory cell at the limit. However, with some components (eg. capacitors), exceeding the limits can cause physical damage to the device.

Linearity

It is important to consider the relationship between the value written to the memory cell (x) and the change in the memory cell's value (y) due to the write operation. Ideally this relationship is both linear and monotonic such that $y = Ax$ where A is constant. We also need to be aware of the difference between idealized components and real-world components since some components have linear idealized models however, in the real world, they are not linear.

However we first need to look at what x and y actually are in practical implementations to fully understand the relationship. x, the value to be written is usually represented as a voltage (unit Volts), a current (unit Amps), a time interval (unit seconds) or a combination of these. y, the value in memory, is usually represented by a charge (unit Coulombs) or resistance (unit Ohms). Since we can not observe or measure the value in memory directly, the circuit used for the read operation needs to first convert the value in memory to a voltage which can be measured by an ADC.

Reading

While digital memory can be read hundreds of times without affecting the value stored in the memory, the same can not be said for analog memory. Every read operation will have some effect on the value stored in the memory cells. We refer to this as a "destructive read" operation however there are different levels of destructiveness. Depending on how the cells and control circuit are designed, this can be anywhere on the following scale:

Minimally destructive: after hundreds of reads, a small degradation from the original value is noticeable. The memory device is designed such that read operations have as little impact as possible. An example of this is magnetic video tapes where the picture quality degrades very slowly with each playback.

Partially destructive: after a single read operation, the value in memory is noticeably different to the original. However the new state of the memory cells is still similar to the original (often a scaled version of the original). There is a high correlation between the original and the new state of the memory cells.

Fully destructive: after a read operation, the state of the memory cells bears no resemblance to the original value. Usually the cells are left in a reset or zero state. An example of this is magnetic core memory from the 1960s which would be reset to zero after every read operation.

Whilst this destructiveness in usually a problem to be solved or minimized when designing memory, we can potentially exploit it for security purposes. It can allow unauthorized read operations to be detectable.

5.2 Practical examples

5.2.1 Capacitors

Capacitors can be used as analog memory devices since they can hold charge and hence the terminal voltage can represent a stored value. In practice, leakage currents mean that the value stored in memory will degrade within seconds however this is still useful for short-term memory. Writing a value to a capacitor cell simply involves passing a current through the capacitor for a controlled amount of time. Since charge = current $\times$ time, the value to be written is converted to current and time values for writing. Reading is done by simply measuring the terminal voltage.

Sample and hold circuits (also know as analog buffers) as shown in figure 5.2 are usually designed using a capacitor as the memory element. These circuits are designed to store an analog voltage for a short period of time. Since the capacitor will slowly discharge, the circuit is designed to minimize the leakage current so that the value can be stored as long as possible with minimal degradation [44]. DRAM also uses capacitors as the storage element and while they are optimized for digital storage, it is still an analog cell.

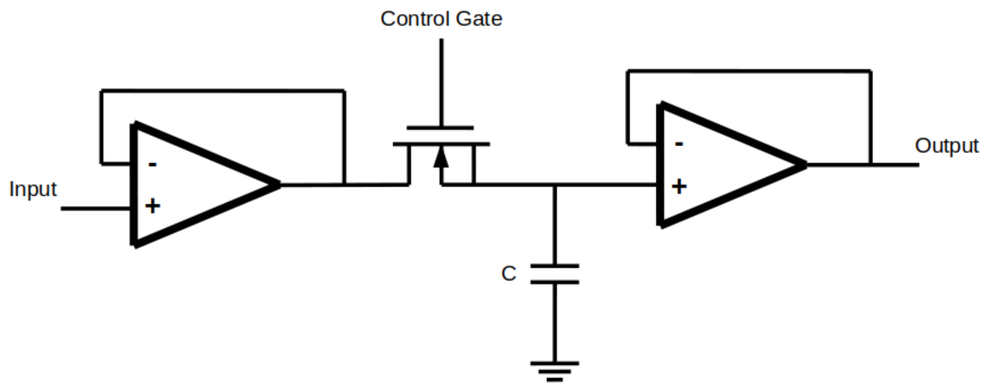


Figure 5.2: Sample and Hold circuit

5.2.2 Floating gate FETs

Floating gate FETs are widely used for digital memory in EPROMs, flash memory, solid state drives and many other memory devices. They operate by storing charge on a 'floating' (electronically isolated) gate within the transistor as shown in figure 5.3. The electric field from the charged gate can then operate the field effect transistor in the same manner as regular FETs. Since the floating gate is electronically isolated, the leakage currents are insignificant and the charge stored on it can remain there for years with minimal degradation. Since no external power is required to keep the charge on the floating gate, these can be used to create non-volatile memory devices [42].

Writing to floating gate FET involves adding or removing charge to the electronically isolated floating gate. A control gate is positioned above the floating gate such that the floating gate is between the transistor channel and the control gate. The floating gate is isolated via thin insulating oxide layers. By applying a relatively large voltage between the channel and control gates, a strong electric field is created across the floating gate which can allow electrons to move across the oxide layer in a process known as 'hot electron injection' [6].

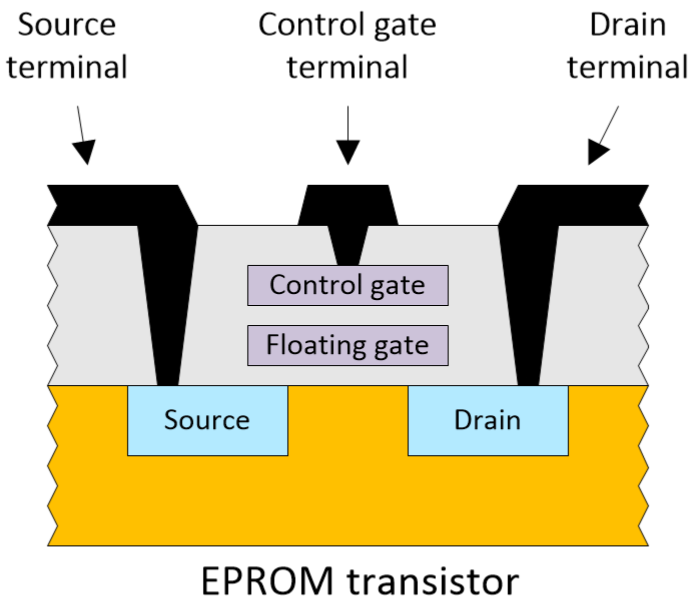


Figure 5.3: Internal structure of a Floating Gate FET [6] (*reproduced with permission*)

It is also possible to use floating gate FETs as analog memory cells [45]. Instead of having just a 'high' and 'low' amount of charge on the floating gate, we can add or remove charge until the gate is at the required value. This value can then be read out by measuring the drain-source current. Adding or removing a given amount of charge to the floating gate is not straight forward due to manufacturing variations so usually a feedback circuit is needed for writing. That is, after adding/removing charge to the floating gate, the drain-source current is read, and any error found is corrected by adding/removing more charge.

5.2.3 Photodiodes/Image sensors

CMOS image sensors are made up of an array of photodiodes along with associated FETs for amplification and switching. Each pixel of the sensor contains a photodiode connected to the gate of a FET as shown in figure 5.4 (the transistor M1 is for resetting the pixel voltage to zero, it is usually in the off state). This structure forms an analog memory cell that stores charge in the same way a capacitor does. When an image is captured, the value of each pixel is stored here as an analog value until it can be processed by the ADC. This is due to the requirement of capturing every pixel at the same instant in time whereas each ADC can only process one pixel's data at a time [46].

When light photons strike the photodiode, change is accumulated across the device (which is also connected to the gate of the FET). The amount of charge is proportional to both the intensity and the duration of exposure to the light. The value is then read using

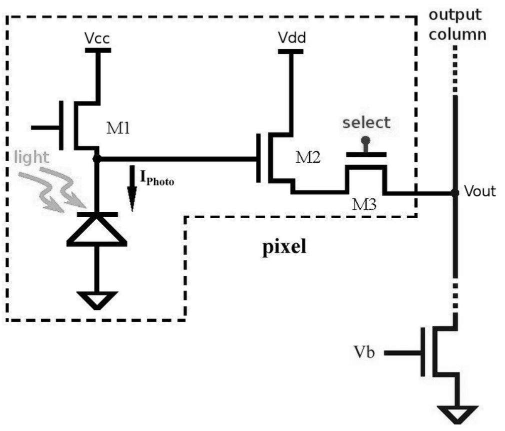


Figure 5.4: Circuit diagram of pixel from a CMOS Image Sensor [7] (*reproduced with permission*)

the same concepts used for floating gate FETs, the drain-source current will be proportional to the charge on the gate.

5.2.4 Memristors

Memristors are a theoretical circuit element that act similarly to a resistor in that they follow Ohm's Law instantaneously. However the resistance varies over time depending on the past current through the device. Current in one direction increases the resistance, current in the other direction decreases the resistance. When there is no current through the device, the resistance remains constant [47]. This can allow memristors to be used as analog memory cells. Using memristors as memory cells is explored further in chapter 6.

5.2.5 Magnetic tape and disks

Magnetic tape was widely used in the 20th century for both analog and digital storage, and is still used for archival storage today. Data is stored by setting the magnetisation of a magnetisable material to a given state. It will then remain in this state for an extended time, without power, where it can be read back at a later time. To allow for a large storage volume, the magnetisable material is constructed on the surface of a long tape which is then mechanically moved over an electronic read/write/erase head [48].

Unlike the other memory technologies discussed, magnetic tape is not split into discrete cells, it is a continuous analog waveform that is written, stored and read from the tape.

Writing to magnetic tape involves applying a suitably large magnetic field to the magnetisable material such that the magnetisation is altered to the level required. The new value will be written additively on top of any existing magnetisation hence an erase operation is usually carried out before writing which sets the magnetisation back to a constant zero level. Reading is done using a magnetic field sensor that can detect the magnetisation level as the tape passes over it.

Magnetic disks operate in a similar way to magnetic tape but with the magnetisable material constructed on a disk instead of tape which allows for quicker random access to data. While focused on digital data storage, there has been work investigating the recovery of erased data from magnetic disks [49]. When data on a magnetic disk is overwritten, there are traces of the previous state left behind. For example, overwriting a zero with a one leaves the magnetisation at 0.95 whereas overwriting a one with a one leaves the magnetisation level at 1.05. The digital read circuitry interprets both of these values as a binary one. However, by using a technique known as magnetic force microscopy, a high resolution image of the magnetisation across the disk can be obtained. This image allows the differences between a 0.95 state and 1.05 state in a cell to be observed, and hence the previous data on the disk can be determined.

5.3 Tamper-evident Memory

We explored using analog memory cells to achieve the requirements of a digital evidence bag as discussed in section 2.5.1. Of the four requirements (tamper-evident, unforgeable, clean, offline), we are primarily looking to achieve tamper-evident memory, however the unforgeable and offline requirements are also considered.

We propose to exploit the additive write and destructive read nature of analog memory cell to design a system that can store digital data in a tamper-evident way. We also need to prevent situations where the memory can be swapped out for a supposedly identical replacement with altered data on it. This is the main security weakness of applications where write-once memory is used. Due to the offline requirement, blockchain based approaches are not suitable since they require network access to get witnesses. The rest of this chapter explores one possible design for how a DEB could be created utilizing analog memory.

For simplicity, we will assume the data being stored in the analog memory cells is a cryptographic hash of the digital data which is stored elsewhere in conventional digital memory. This means the data going into the analog memory cells is of a fixed length.

5.3.1 Additive writes

The simplest way to store digital data in a analog memory cell is to assign a set ‘step size’ S to a digital ’1’ and nothing to a digital ’0’. When writing a ’1’, the value in the analog cell is increased by S and writing a ’0’ results in no change to the memory cell. Assuming the memory cell(s) have been erased to a known zero value before writing, it is straight forward to read the data out and convert back to digital. This is almost exactly how conventional digital memory works and we have explored the security features of measuring the analog output of the digital memory array in chapter 4.

So we move on to looking at what happens if a memory array is written to multiple times between erasures. To explore the principle of operation, we assume an erase will set all cells to zero and we assume that the noise level is negligible. After a given number of writes, the value of a memory cell is $S \times x$ where x is the number of digital '1's that have been written to that cell. For example, we write the following four 8 bit numbers to a 8 cell analog memory array.

01010101
01101111
01100110
01000001

After this data has been written to the memory array, the values in the eight cells will be:

$$0, 4S, 2S, 1S, 1S, 3S, 2S, 3S$$

After reading this analog data out, we are unable to reconstruct the original digital data from it. All we know is that the Most Significant Bit (MSB) of each value written is a '0' and, if we know only four values were written, then the 2nd MSB of each value is a '1'. The other noteworthy point is that assuming $S > 0$, it becomes impossible to over-write previously written data. The only way to remove data from the memory array is to perform a erase operation.

However, for the memory array to be useful, we need to be able to recover the original digital data written to it. To achieve this we look at the algorithms used in CDMA.

5.3.2 CDMA

Code Division Multiple Access (CDMA) is a channel access method commonly used in radio communications to partition the channel so it can be used simultaneously by a number of different users. It is used in a number of mobile phone standards and the GPS satellite navigation system [50]. For our purpose we will only look at synchronous CDMA which is based on orthogonal codes.

In the context of CDMA, an orthogonal code is an array of digital values (1 and -1) that when multiplied with another orthogonal code, then summing the result gives a total of zero [51]. This essentially means that there is no correlation between any two orthogonal codes. For example, when the two orthogonal codes (1, -1, -1, 1) and (1, -1, 1, -1) are multiplied together the result is (1, 1, -1, -1), which if all the values are summed together gives zero.

Synchronous CDMA operates by assigning each user an orthogonal code to spread their data with. The number and length of the orthogonal codes depends on the number of user channels required. Spreading the data is done by replacing each bit of the data with a copy of the orthogonal code. As shown in the example below, the orthogonal code is inverted depending on the value of the input bit.

Input data: 0110
Orthogonal code: (1, -1)
Data after spreading: (1, -1) (-1, 1) (-1, 1) (1, -1)

After the data for each user has been spread with the user's unique orthogonal code, it is all added together into a single signal which is then transmitted over the air. The example below continues the example above with a 2nd user's data of 0101 and orthogonal code of (1, 1)

User 1 data after spreading: 1, -1, -1, 1, -1, 1, 1, -1
User 2 data after spreading: 1, 1, -1, -1, 1, 1, -1, -1
Add these to signals together: 2, 0, -2, 0, 0, 2, 0, -2

At the receiving end, the same signal (ignoring noise) is received by all users who then have to use their orthogonal code to extract the data intended for them. The signal is first split into sections the same length at the orthogonal codes. Then the receiver calculates the vector dot product of each section dotted with the user's orthogonal code. And finally the result is converted back to binary by taking the sign of each number. The example below shows the receiving procedure for extracting user 1's data used their orthogonal code (1, -1).

Received signal: (2, 0), (-2, 0), (0, 2), (0, -2)
Dot product with user 1's orthogonal code: (2, 0).(1, -1), (-2, 0).(1, -1), (0, 2).(1, -1), (0, -2).(1, -1)
This calculates to: 2, -2, -2, 2
Convert to binary to get: 0110

This process allows each user to recover their data from the received signal after all user data has been added together. This process is also known for its noise immunity, in that it can operate with an acceptable BER in a very noisy environment.

CDMA applied to analog memory

Building on the additive write process described in section 5.3.1, we aim to use CDMA techniques to make the original data recoverable after reading from the memory array. To do this we first need to generate some orthogonal codes and spread the data with them before writing it to the memory array. This is a digital process which will increase the number of bits to be written. Since we are writing 1 bit per analog cell, this will require more memory cells. The number of cells required is equal to the number of bits in the original data multiplied by the number of bits in the orthogonal spreading codes. For example we will use the four 8 bit numbers from section 5.3.1 and spread them with the four 4 bit spreading codes shown below.

1111
1010
1100
1001

After spreading, each piece of data will be $4 \times 8 = 32$ bits long. The four pieces of data to be written to the memory array are shown below:

```
1111  0000  1111  0000  1111  0000  1111  0000
1010  0101  0101  1010  0101  0101  0101  0101
1100  0011  0011  1100  1100  0011  0011  1100
1001  0110  1001  1001  1001  1001  1001  0110
```

Each of these is then written to the analog memory array (of 32 cells). When each one is written, it will be added to the existing data in memory. Assuming the cells are all at zero to begin with, the value in the cells after writing all four data sets will be:

$$R = (4\ 2\ 2\ 2\ \ 0\ 2\ 2\ 2\ \ 2\ 2\ 2\ 4\ \ 3\ 1\ 1\ 1\ \ 3\ 3\ 1\ 3\ \ 1\ 1\ 1\ 3\ \ 2\ 2\ 2\ 4\ \ 1\ 3\ 1\ 1)$$

(assume step size, $S = 1$ for simplicity)

After reading this combined data back out from the memory array, each of the four individual chunks of data written can be recovered if the orthogonal codes are known. We first need to scale the data to account for the fact that we have excluded negative values so far. We need to subtract half the number of writes that were done. So in the example above, four pieces of data were written to the memory so we need to subtract 2 from the data read out.

$$R_{adj} = ([2\,0\,0\,0]\ [-2\,0\,0\,0]\ [0\,0\,0\,2]\ [1\,-1\,-1\,-1]\ [1\,1\,-1\,1]\ [-1\,-1\,-1\,1]\ [0\,0\,0\,2]\ [-1\,1\,-1\,-1])$$

We can then calculate the vector dot products with the orthogonal code(s).

$$R_{adj} \cdot [1\ 1\ 1\ 1] = (2\ \ -2\ 2\ \ -2\ 2\ \ -2\ 2\ \ -2)$$

$$R_{adj} \cdot [1\ \ -1\ 1\ \ -1] = (2\ \ -2\ \ -2\ 2\ \ -2\ \ -2\ \ -2\ \ -2)$$

$$R_{adj} \cdot [1\ 1\ \ -1\ \ -1] = (2\ \ -2\ \ -2\ 2\ 2\ \ -2\ \ -2\ 2)$$

$$R_{adj} \cdot [1 \;\; -1 \;\; -1 \;\; 1] = (2 \;\; -2 \; 2 \; 2 \; 2 \; 2 \; 2 \;\; -2)$$

Then take the sign of each number to get the original data:

$$\begin{array}{c} 01010101 \\ 01101111 \\ 01100110 \\ 01000001 \end{array}$$

By using CDMA techniques, we can easily recover all the data that was written to the memory array, however it comes at the cost of a greater number of memory cells required. There is also the issue of bit errors to deal with. Since CDMA is known for its good noise immunity so we expect the memory system described above to be less prone to bit errors. However, the types of errors and the rate of them depend on the technology used to implement the analog memory cells.

PN codes

Pseudo-random binary sequences are sequences of binary digits that satisfy the tests for statistical randomness but are generated deterministically. They are used to create pseudo-noise codes (PN codes) which are used in CDMA radio interfaces to make signals look like background noise and not interfere with other users signals. PN codes are generated by deterministic linear shift registers so while they appear random, they can easily be recreated if the initial conditions are known.

There are a number of tests for statistical randomness, the more common ones are listed below [52]. PN codes are designed such that they pass these tests.

- Frequency test: The relative frequencies of '0' and '1' in the sequence should be approximately 50% each.
- Run test: Count the number of runs of the same digit within the sequence. It should be approximately $\frac{(n+1)}{2}$ for a sequence that is n digits long.
- Linear complexity test: Find the length of the shortest linear feedback shift register (LFSR) that can generate the sequence. The longer this LFSR is, the more random the sequence is.

PN codes are designed to be uncorrelated with both other PN codes and shifted versions of themselves. By encoding two sets of correlated data with two unique PN codes, we can make them appear uncorrelated. Going back to our analog memory array, before we write the data to the memory array, we first encode it with a known PN code. For reasons explained later, this is used to ensure the data stored in the memory is uncorrelated with other data.

5.3.3 Read process

To read an analog memory cell we need to measure the value in the cell and convert it to a digital number for processing. This is likely to be done with an ADC of some sort which will introduce a quantization error into the result. We need to ensure the ADC resolution is large enough to make any quantization errors small enough to be negligible. After every cell in the memory array has been read, we will have an array of digital numbers.

If writing to the memory is additive only, we also need to provide an erase function to return each cell to a zero value. If not, the memory effectively becomes write-once since there is no way to reuse it.

No erase

But first we should also consider what happens if there is no erase function. This creates a similar situation to what write-once memory is. However since there are multiple bits of data in each cell, the memory can be written to many times until the cells eventually reach a 'full' state. At this point no more writing is possible and a new memory device will need to be acquired for the next use.

This creates some level of tamper evidence since it is impossible to erase or alter the data after it has been written. In that sense, it operates very similarly to write-once digital memory and has the same security benefits and weaknesses. The main security weakness being that an attacker can read the data, alter it, then rewrite it to a new device (while destroying the original). The device fingerprinting explored in chapter 4 can offer some protection against this attack however it relies on keeping good records on memory devices, their fingerprints and where they have been used.

Destructive Read

Next we look at what happens if we design the read process to be a total destructive read. That is, reading the memory also erases it in a single operation. This creates some level of tamper evidence since just reading the data creates an easily observable change in the data.

However an attacker can easily re-write the data to memory after they have read it. They can also alter the data before it is re-written. Other than some random noise, the re-written data will likely be indistinguishable from the original data.

Before moving further, we need to look at the system so far from a mathematical point of view. We have an array of memory cells, with each cell storing one value, which is a number greater than zero.

M represents the current state of the memory array, where $M(i)$ is the value in the ith memory cell in the array.
W is the data to be written to the memory array (after CDMA encoding) which is a

‘digital’ value, in that only has two states, equal to zero or equal to the step size, S
R is the output from a read operation where $R(i)$ is the value read from the ith memory cell in the array.

We also define a single ‘memory cycle’ here as being: multiple writes followed by a read and erasure. At the start and end of each cycle, the memory array is in the erased (zero) state. Each cycle outputs one read value for each cell in the memory array. There is no theoretical limit of the number of cycles a given memory device can endure. However the technology used to implement the memory array will impose limits on the number of cycles before it wears out.

The following is a step-by-step walk through of a single cycle.

At the beginning: $M(i) = 0$
After the 1st piece of data has been written: $M(i) = W_1(i)$
After the 2nd piece of data has been written: $M(i) = W_1(i) + W_2(i)$
...
After the final piece of data has been written: $M(i) = W_1(i) + W_2(i) + .. + W_n(i)$
And finally at the end after a destructive read has taken place:
$M(i) = 0$ and $R(i) = W_1(i) + W_2(i) + .. + W_n(i)$

In order to differentiate the read data from different cycles, we use k to represent the cycle number, so $R_k(i)$ is the read data from the kth cycle.

Practical example

The circuits in figure 5.5 show how different read operations can be achieved on a capacitor based analog memory cell. Similar circuits can be designed for other memory cell technologies. Memristor cells are discussed further in the next chapter.

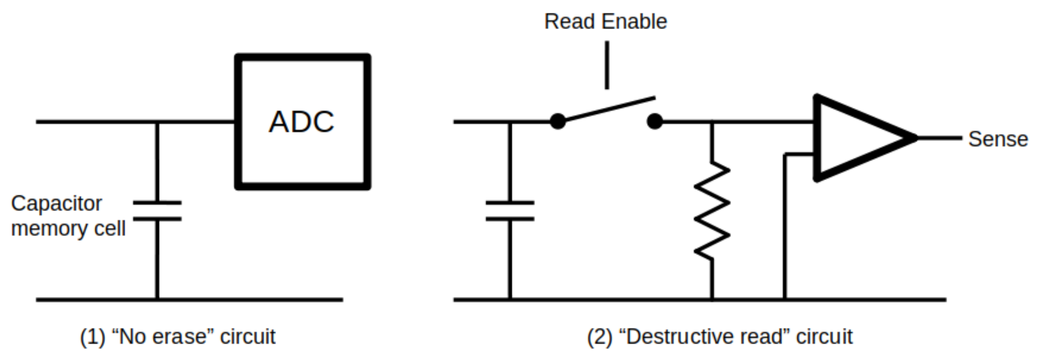


Figure 5.5: Example circuit for a memory cell with destructive read

The circuit in (1) uses a conventional ADC to measure the voltage directly which minimizes any change to the state of the memory cell. While ideally the ADC input has a zero current, real ADCs all have a small current on the input which would cause a small change to the charge on the capacitor. In a practical sense, a large array of cells would need to share a single, or smaller group of ADCs so some multiplexing would need to be included as well.

The circuit in (2) uses a timer to measure the time the capacitor takes to discharge. A timer is started when the read enable switch is closed and the capacitor starts to discharge through the resistor. Once the voltage comparator has sensed the voltage reaching zero (within a tolerance), the timer is stopped. Since the time taken to discharge is proportional to the voltage in the capacitor, this time measurement will represent the cell's value. Since the capacitor is left in a discharged state after the read operation, this is a fully destructive read.

It is important to note that these circuits are simple examples. There are other ways these circuits can be designed to optimize and achieve different outcomes.

5.3.4 Read detection

Here we look at how to detect any read operations that are done on the memory. The read, alter and re-write issue discussed in the previous section could potentially be stopped by detecting any unauthorized read operations that were performed. The presence of a read operation that is unaccounted for would suggest that the data may have been tampered with.

Loosely inspired by hashchains, we pose the question, what if each read operation could be linked to the previous and next read operations via some random number? This would create a chain where any unauthorized read operations would be detectable as a missing link in the chain. Figure 5.6 shows the difference between a valid and invalid chain.

To achieve this in practice, each set of read data needs to contain two numbers, one to link forward and one to link backwards. These numbers can not be considered secret since they need to be visible in order to verify the chain. Since an attacker may know the previous read data's link numbers, the system needs to be designed so that the attacker has no control over what the link number will be (for example, they could be chosen at random by the memory device) for the next read operation.

5.3.5 Residue

Here we look at what happens if instead of erasing each memory cell to zero, a small 'residue' value is left in every memory cell. An erase operation would reduce the value in each cell down to the residue value. For simplicity we'll start by assuming the residue value is chosen

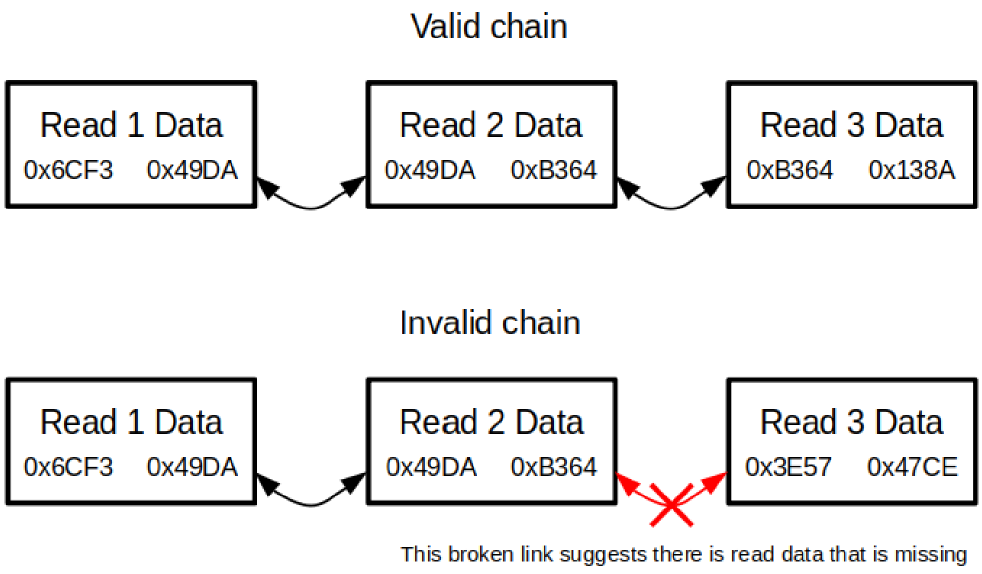


Figure 5.6: By chaining read operations together we can detect missing read operations

at random such that it is between zero and the value in the memory cell (a different residue value is randomly chosen for each cell).

We define $T(i)$ to represent the residue value for the ith cell.

The read/erase operation then results in the following happening:
The data read out is: $R(i) = M(i) - T(i)$
The state of the memory cell after the read operation is: $M(i) = T(i)$

This creates a non-zero 'erased' state between cycles so we need to take this into account as well. The following is the step-by-step walk through of the kth cycle with residues.

At the beginning: $M(i) = T_{k-1}(i)$ (Memory starts with the residue from the previous cycle)
After the 1st piece of data has been written: $M(i) = T_{k-1}(i) + W_1(i)$
After the 2nd piece of data has been written: $M(i) = T_{k-1}(i) + W_1(i) + W_2(i)$
...
After the final piece of data has been written: $M(i) = T_{k-1}(i) + W_1(i) + W_2(i) + .. + W_n(i)$
And finally at the end after a destructive read has taken place:
$M(i) = T_k(i)$ and $R_k(i) = T_{k-1}(i) + W_1(i) + W_2(i) + .. + W_n(i) - T_k(i)$

The important thing to note here is that $R_k(i)$ contains both the current and previous residue values. This should cause $R_k(i)$ to correlate with $R_{k-1}(i)$ and $R_{k+1}(i)$ which can potentially give us a method for detecting unauthorized read/erase operations. This can be more easily seen if we look at the data from many cycles as shown below.

$$R_{k-1}(i) = T_{k-2}(i) + W_{all}(i) - T_{k-1}(i)$$

$$R_k(i) = T_{k-1}(i) + W_{all}(i) - T_k(i)$$

$$R_{k+1}(i) = T_k(i) + W_{all}(i) - T_{k+1}(i)$$

(where $W_{all}(i)$ represents the sum of all the write data for the given cycle)

From this we can expect $R_k(i)$ to correlate with both $R_{k-1}(i)$ and $R_{k+1}(i)$ however there should be no correlation between $R_{k-1}(i)$ and $R_{k+1}(i)$. The strength of the correlation depends on the relative magnitude of each of the components which is discussed further in the next section. So far we have shown that by correlating the residue values we can potentially detect missing cycles. If there is no correlation between supposedly sequential blocks of read data, then it would suggest that an unauthorized read/erase has occurred in-between.

Residue circuit

With some small modifications to the circuit from figure 5.5, a practical example of how a memory cell with residue would be built is shown in figure 5.7. First a residue value is chosen and applied to input T (possibly from a random number generator but there are other options for this). Then a timer is started along with the read enable switch being closed. Once the voltage comparator has sensed that the voltage has dropped below the residue value, the timer is stopped. The read enable switch is also opened at this time.

There are two different scenarios this circuit operates in. The main one is when the value in the memory cell is greater than the residue value. In this case the output from the timer will represent the value in the cell minus the residue value.

For the case $M > T$:
The value read out: $R = M - T$
The state of the memory cell after reading: $M = T$

The other case we need to consider is when the value in the memory cell is less than the residue value. In this case the voltage comparator will sense immediately and the timer will stop at zero.

For the case $M < T$:
The value read out: $R = 0$
The state of the memory cell after reading: $M = M$ (no change to the memory cell)

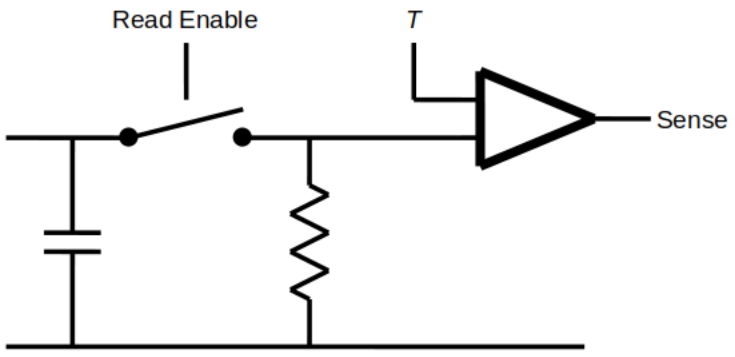


Figure 5.7: Example circuit for a memory cell with destructive read and residue

We need to be aware of this case when selecting the residue value.

5.3.6 Other issues

Unwanted correlations

It is important that the data from each cycle is uncorrelated since this could produce some false positive matches. $R_{k-1}(i)$ and $R_{k+1}(i)$ will only be uncorrelated if the write data within them is uncorrelated. This is a problem since regular uncompressed and unencrypted digital data is often full of patterns that cause strong correlations between data of the same type.

To ensure the data is uncorrelated, we encode each block of data with a PN code before writing it to the memory array. It is important to note that the same PN code is used for all the writes on a given cycle in order to make the synchronous CDMA orthogonal codes work. A new, unique PN code is used for each new cycle of the memory array. This ensures that the data within the read output from each cycle is uncorrelated to the data in earlier cycles.

The residue values for each cycle also need to be chosen in a way that they are uncorrelated to each other.

Magnitude of the correlation

There are three components that make up the read output: $T_{k-1}(i)$, $W_{all}(i)$ and $T_k(i)$. The relative size of each of these determines how strong the correlation will be. Only one of the residue values is relevant for each correlation calculation.

The size of $W_{all}(i)$ is as follows:

- Minimum W_{all} is 0 (cell written with only binary '0's)

- Maximum W_{all} is $S \times N$ (cell written with only binary '1's)
- Average W_{all} is $\frac{S \times N}{2}$ (since the data in W_{all} is encoded with a PN code which we know is statistically random, ie. there will be an approximately equal number of '1's and '0's)

N, the number of writes to the memory array, can vary each cycle from 0 up to a set maximum (which is set by the physical limits of the technology used to implement the memory cells). N is also limited by the number of orthogonal codes available for CDMA spreading. Hence the correlation of sequential blocks of read data needs to work effectively for all values of N.

We get to choose the maximum of $T(i)$ and we will set the minimum of $T(i)$ to zero since any greater value would mean the memory cells never get erased to zero and hence the lower part of the cells is effectively unused and wasted. We'll assume the values for $T(i)$ are evenly distributed so the average will be half of the maximum. It is important to note that the min, max and average will be the same for all cycles (values of k). ie. the min and max of $T_k(i)$ is the same as $T_{k-1}(i)$.

Hence the step size S determines the size of $W_{all}(i)$ (along with the number of writes) and the max(T) setting determines the size of $T(i)$. We need to choose these parameters so that the magnitude of the correlation is large enough to be detectable. How large is large enough will be determined though simulations and experiments.

$M < T$ errors

As briefly mentioned in the previous section, there is a special case to deal with when $M < T$. The output from the read operation is summarized below:

$$R_k(i) = \begin{cases} T_{k-1}(i) + W_{all}(i) - T_k(i) & \text{if } M(i) > T_k(i) \\ 0 & \text{if } M(i) < T_k(i) \end{cases}$$

Where $M(i) = T_{k-1}(i) + W_{all}(i)$

The $M < T$ case causes an error, both in the reading of the original data as well as in the correlation between blocks of read data from sequential cycles. Synchronous CDMA has some level of error immunity built in to it however a conventional error correction code would still need to be used if bit errors in the digital data are unacceptable. The correlation will tolerate some errors with the result being a reduced correlation strength (relative to the number of errors). Hence the $M < T$ case is tolerable so long as it only occurs rarely. Since the average of W_{all} is $\frac{S \times N}{2}$ and W_{all} is the main component of M, we expect this error to only be a problem for small values of N.

First we look at what happens when $N = 0$ (that is, no data has been written to the memory array). In this case $W_{all}(i) = 0$ and hence $M(i) = T_{k-1}(i)$ when the read operation takes place. The $M < T$ case will occur on approximately half (50%) of the cells. Hence approximately half of the values in $R_k(i)$ will be 0 and won't contribute to the correlation. The $N = 0$ case becomes more challenging when we consider multiple sequential cycles, each with $N = 0$. After one $N = 0$ cycle, the state of the memory array is $M(i) = \min(T_k(i), T_{k-1}(i))$. If a second $N = 0$ cycle happens after that, $M < T$ will occur on approximately two-thirds (66.7%) of the cells. After a third $N = 0$ cycle happens, $M < T$ will occur on approximately three-quarters (75%) of the cells, and so on. This will cause the correlation to slowly decrease as more $N = 0$ cycles happen.

For other values of N, the average value of $W_{all}(i)$ increases with N, the number of writes since the average of W_{all} is $\frac{S \times N}{2}$.

$$\text{average}(M) = \text{average}(T_{k-1}) + \text{average}(W_{all})$$

Hence the average of M will be greater than the average of T_k for all values of $N \geq 1$ and $S \geq 1$. This means the rate of $M < T$ errors will be under 50% whenever $N \geq 1$.

Source of the residue values

So far we have just assumed the residue T comes from a random number generator. There are other options and considerations to take into account here. First we look at what the characteristics and requirements are for the source of the reside values.

- Not secret. It is possible to recover the residue values from the read data $R(i)$ so we have to assume that any attacker would know the residue values of all past cycles.
- Unpredictable. The residue values of future cycles need to be unpredictable to an attacker, even when they know all past residue values. This ties in with the unclonable requirement below.
- Unique and never repeated. The residue values for a given cycle must never be reused on another cycle. If they are repeated, all the data between the two cycles with repeated residue values could be maliciously deleted, and this deletion would be undetectable.
- Uncorrelated. The residue values for each cycle must be uncorrelated since we use correlation to detect missing cycles. Any correlation between residue values from different cycles could cause maliciously deleted cycles to go undetected.
- Unclonable. The system that creates the residue values should be unclonable. This helps in preventing an attacker from creating a fake memory device that impersonates the original one.

A common method used in electronics for creating random numbers is to use a pseudo-random number generator (PRNG) which uses a deterministic algorithm to create an output that passes the tests for statistical randomness (ie. it appears random). The input to these algorithms is known as the seed value and using the same seed value twice will produce the same random numbers. Since PRNGs are deterministic, their output is predictable if the seed is known, which in our case can be inferred from past residue values. As a result, a simple PRNG is not suitable for creating the residue values.

Another method to create random numbers is to measure an some external physical phenomenon that is known to exhibit random properties. Whilst more complex to build, the output from this method is unpredictable. This can be combined with PRNGs to create whats known as "cryptographically secure pseudorandom number generators". These are potentially a suitable source of the residue values since all the requirements except unclonability are meet. Unclonability could be achieved by another part of the system unrelated to the residue values (for example, with a digital fingerprint) in which case, the unclonability requirement is not critical here.

Physically unclonable functions (PUFs as discussed in section 3.6.5) could also be useful here. By using a PUF as the seed to a PRNG, the unclonable requirement is meet. The output would be predictable if the behavior of the PUF is known, however this can be kept secret since it cannot be inferred from past residue values. Using a cryptographic challenge-response could also be an option if it is preferred not to use the manufacturing variances based PUF approach.

5.4 Simulation

In order to test the idea of correlating sequential blocks of read data, a simulation was carried out in MATLAB using an array of floating point numbers to represent the memory array. We also experimented with the size of S and T for a number of different writes per cycle (N).

The CDMA spreading codes used were 256 bits in length and the input data was 256 bits in length which meant we required a memory array of 65,536 analog cells. No limit was placed on the maximum value that could be stored in the cells, however the number of writes per cycle (N) is still limited by the number of available spreading codes (256 in this case).

5.4.1 Process

To begin with an array of floating point numbers was created to simulate the memory cell array, called $M(i)$. Each cell was initialized to a random number with the range of the residue values. This represents the memory array in the erased state.

Writing

The write data used for testing was the output of a SHA256 hash of some basic text data. It was then spread with an 256 bit orthogonal code unique within the current memory cycle. The spread data was then encoded with a PN code to ensure it was uncorrelated with other data. The same PN code was used for all write operations within the same memory cycle and a new one was used for each new cycle. The encoded data was then added to the memory array by adding the step size, S to the cells that corresponding to a binary '1'. Since this is an analog operation, a small amount of random noise was also added to represent the imperfections of the analog system.

The write process was repeated a number of times (N) for each cycle.

Reading

Reading was done once at the end of each cycle. First a residue array $T(i)$ is created by a random number generator. The numbers being evenly distributed between the residue min and max settings. Then the output data $R(i)$ is calculated by computing $R(i) = \max(M(i) - T(i), 0)$. The memory array is also reset to the erased condition by setting $M(i) = \min(T(i), M(i))$. The output data $R(i)$ is then saved before starting the next cycle on the memory.

Each time the simulation was run, a total of 20 cycles were simulated, giving 20 sets of read data $R_k(i)$ for analysis (ie. k ranges from 1 to 20).

Analysis

The read data $R(i)$ from each cycle was then decoded and compared to the original write data. From this the bit error rate (BER) was calculated.

The correlation between every combination of two sets of read data was then calculated. ie. $R_1(i)$ was correlated with every other $R_k(i)$, then $R_2(i)$ was correlated with every other $R_k(i)$ and so on. This results in a table of correlation data. One row of this ($R_{10}(i)$) is shown in figure 5.8 where the (negative) correlation with $R_9(i)$ and $R_{11}(i)$ can be seen.

The impact of tampering

A malicious attacker wishes to tamper with (edit or delete) the data from the memory array without being detected. The additive write and destructive read system enforced by the hardware means that the attacker must do a read operation to delete the data from the memory. After the destructive read, they can then edit and rewrite the data to the memory. To detect this kind of attack, we must be able to detect that a destructive read operation has taken place.

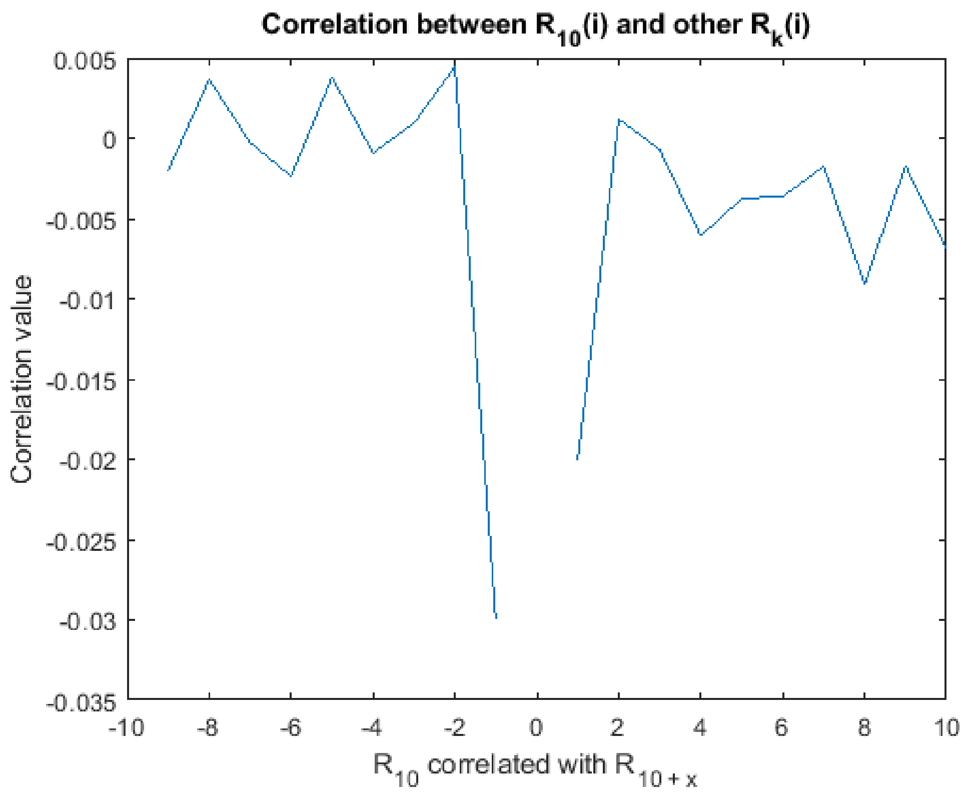


Figure 5.8: Correlation results for $R_{10}(i)$

Using figure 5.8 as an example, we consider what would happen if R_{11} was a read operation performed by an attacker attempting to go undetected. In this case, the verifier would calculate the correlation between R_{10} and R_{12} (since R_{11} has been deleted, they are assuming that R_{12} is R_{11}). There is no correlation between R_{10} and R_{12} which shows that at least one R_k is missing and the data has been tampered with.

In the following section we see that this concept applies to all R_k. There is only a correlation between R_k from sequential cycles and if any R_k are deleted, the lack of correlation will show that there is a missing R_k.

5.4.2 Results

To start with we set $S = 10$, residue, T_k between 0 and 20 and then write to the memory 255 times per cycle ($N = 255$). The random noise added when writing has a standard deviation of 1. Ten cycles were simulated resulting in ten $R(i)$ data arrays. In figure 5.9 the correlation results for each $R_k(i)$ have been overlaid and aligned so that the $R_{k-1}(i)$ and $R_{k+1}(i)$ correlations line up. From that it can be seen the correlation has a greater magnitude for the $R_{k-1}(i)$ and $R_{k+1}(i)$ cases, where as there is a much lower correlation with all other $R(i)$. In this case the bit error rate (BER) after reading the hashes was 0.14%. We use this result as a starting point to compare the effects of changing the parameters of the simulation.

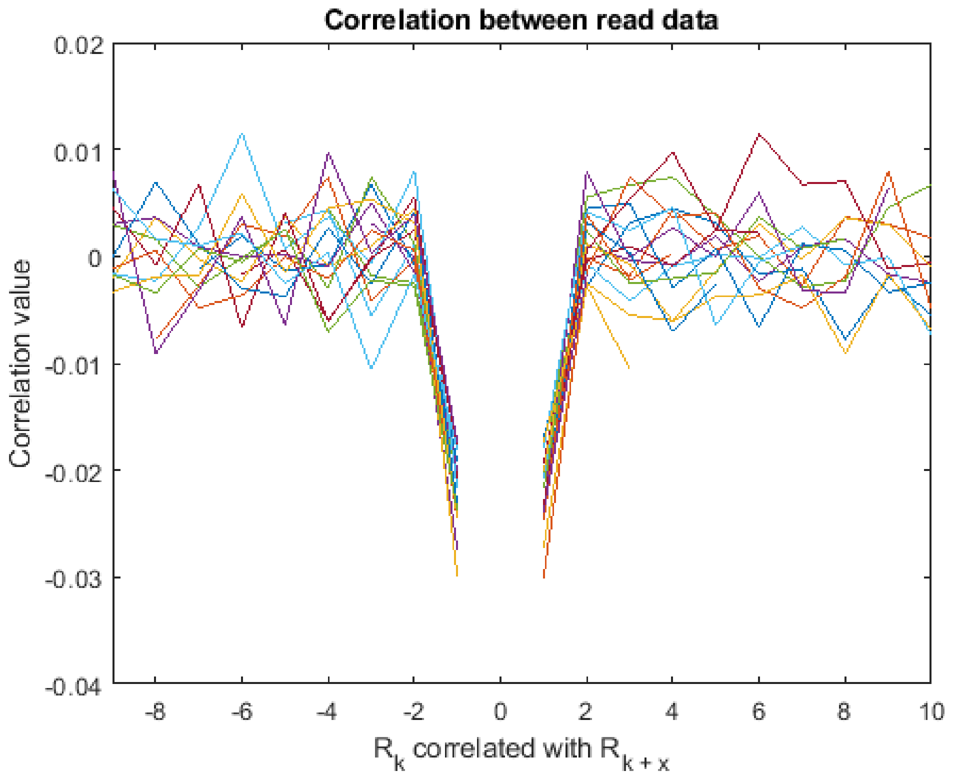


Figure 5.9: Correlation results after writing 255 hashes per cycle to the memory array (each line represents one value of k for R_k)

5.4.3 Step size

Here we look at the effects of the step size (S) on the system. Whilst the simulation has no limit on the maximum value stored in the memory cells, the memory elements used in practice will have limits on the maximum value. Larger step sizes will cause this limit to be reached quicker and thus will reduce the amount of data that can be stored before the memory is ‘full’. The technology used to implement the analog memory cells also needs to be considered when choosing the step size since smaller step sizes makes the system more susceptible to errors caused by noise and quantization errors (if the quantization level of the ADC isn’t reduced accordingly).

The simulation was run with step sizes of 1, 5, 10 and 20 and the correlation results are shown in figure 5.10 and the BER are shown in table 5.1. It can be seen that larger step sizes reduces the BER at the expense of make the correlation with sequential cycles harder to detect. This is expected since the step size affects the ratio between the data and residue within the read output.

There are two competing aims here, one is to ensure the correlation between the sequential cycles is at a level that can be detected, and the second is to minimize the BER so that the number of extra error correction bits required is minimized. By looking at the correlation results in figure 5.10 it can be seen that there is little difference in the correlation between sequential cycles and the correlation between non-sequential cycles in the $S = 20$ case. In the other three cases considered there is a noticeable difference in the correlation between sequential and non-sequential cycles, although it is a small difference in the $S = 10$ case.

The BER is relativity low in all cases except the $S = 1$ which suggests that best case of the four overall is when the step size is 5. The step size $S = 5$ creates a ratio of the $\max(T)$ being 4 times greater than the step size.

Step Size	Bit Error Rate	Max cell value
1	35%	244
5	2.5%	858
10	0.005%	1658
20	0%	3352

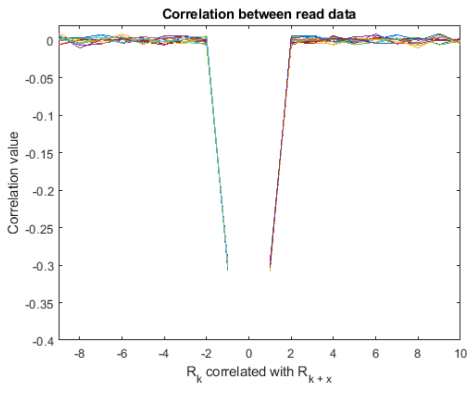
Table 5.1: Bit Error Rate and Max cell value for different step sizes

5.4.4 Residue range

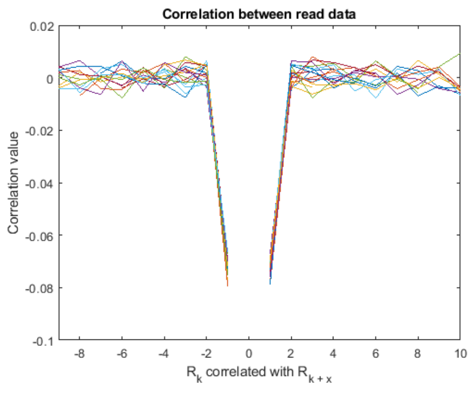
The residue values, T_k, were created by a random number generator with a number of different range settings. The simulation was run with ranges 0-1, 0-10, 0-20 and 0-50 and the correlation results are shown in figure 5.11, along with the BER results in table 5.2. The random number generator was configured such that the outputs were equally spread across the given range.

Residue range	Bit Error Rate	Max cell value
0-1	0%	1679
0-10	0%	1658
0-20	0.005%	1658
0-50	3.5%	1748

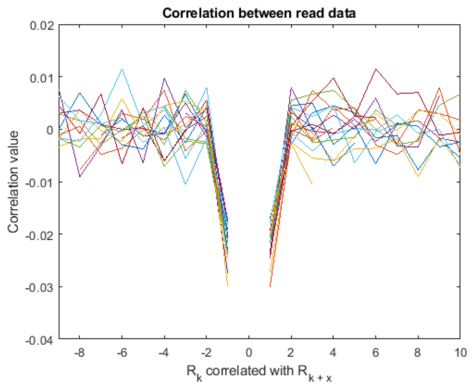
Table 5.2: Bit Error Rate and Max cell value for different residue ranges



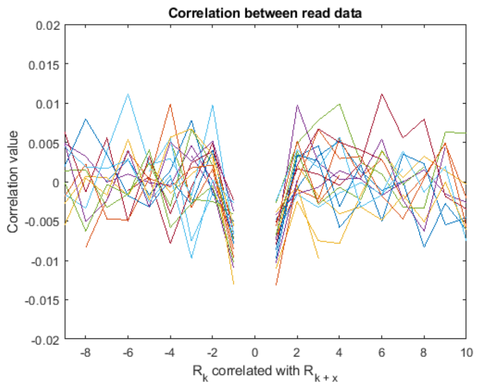
(a) Step size of S = 1



(b) Step size of S = 5



(c) Step size of S = 10



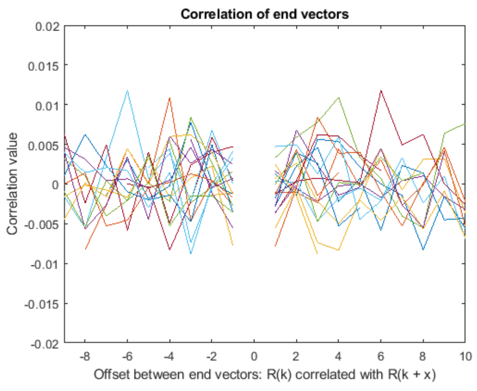
(d) Step size of S = 20

Figure 5.10: Effect of changing the step size

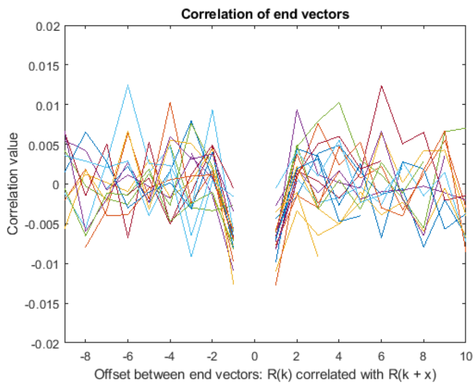
It can be seen that increasing the range of the residue values causes the correlation between each read cycle to become stronger at the expense of a slightly increased BER. Like with the step size, this is the expected result since the size of the residue values also affects the ratio between the data and residue within the read output. The best trade-off, where the correlation is detectable and the BER minimized, occurs with a reside range of 0-20, although the BER with the reside range of 0-50 is also small. These two cases represent a ratio of $\max(T)$ being 2 and 5 times greater than the step size S.

5.4.5 Writes per cycle

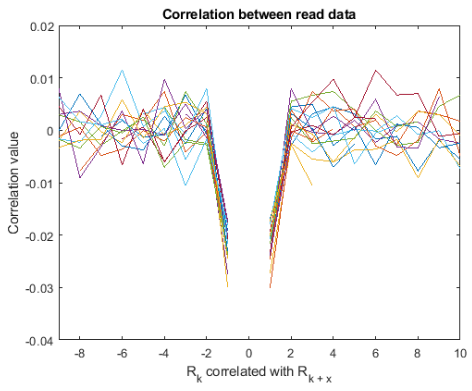
The number of hashes written to the memory per cycle (N) will likely vary with each cycle. Hence the system needs to operate correctly for all possible values of N. The minimum $N = 0$ occurs when the memory is read before anything has been written to it which does not appear to have any useful purpose, and causes problems with $M < T$ errors as discussed



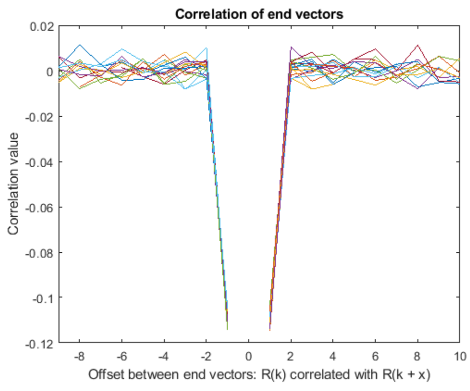
(a) Residue range of 0 to 1



(b) Residue range of 0 to 10



(c) Residue range of 0 to 20



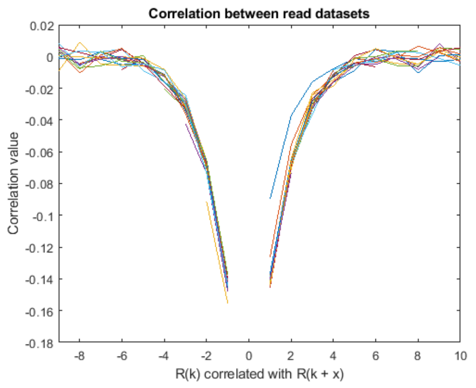
(d) Residue range of 0 to 50

Figure 5.11: Effect of changing the residue range

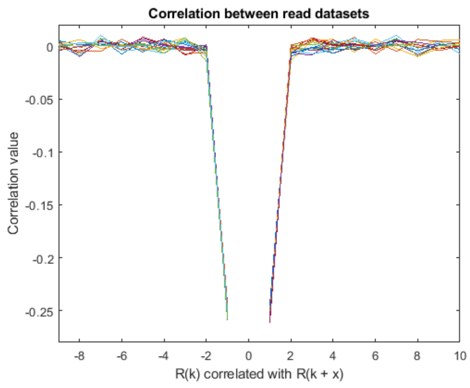
earlier. So we consider the minimum for N to be 1 and the maximum to be equal to the number of orthogonal spreading codes, which is 255 in this simulation. In real memory cells, the maximum N may also be limited by the maximum value that can be stored in the cells, however in this simulation there is no maximum value for each cell.

Figure 5.12 shows the effect on the correlation results for different values of N and the corresponding BER results in table 5.3. These results suggest that the number of writes per cycle doesn't have a significant impact on the BER and the impact on the correlation between sequential read cycles is also relatively small for $N \geq 10$.

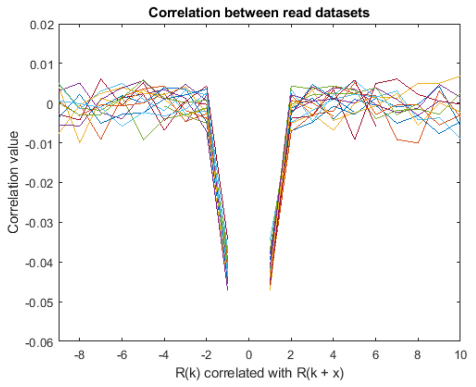
However in the $N = 1$ case it can be seen that there is a small, but noticeable, correlation between non-sequential read cycles. This will cause a significant problem when trying to detect unauthorized read operations. We believe the cause of this is too many cells with $M < T$ when the read operation takes place. When this happens, residue values from previous cycles are left in the cell since the read operation leaves the cells unchanged if $M < T$. When they do eventually get read out after further cycles, they cause small correlations between $R_k(i)$ and $R_{k-2}(i)$, $R_{k-3}(i)$.



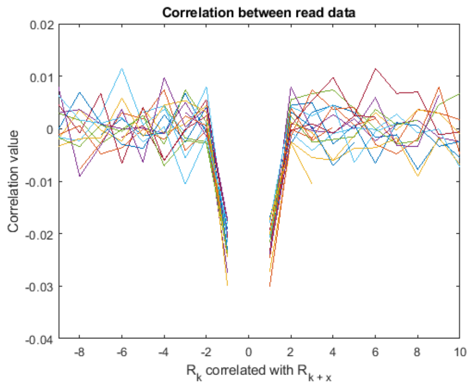
(a) 1 write per cycle



(b) 10 writes per cycles



(c) 127 writes per cycles



(d) 255 writes per cycles

Figure 5.12: Effect of changing the number of writes per cycles

5.4.6 Limitations

Lack of manufacturing variances and defects

One aspect that we are unable to look at in simulations is the effect of the manufacturing variances and defects. From section 3.6.2, where we outlined the properties of a digital fingerprint, we know that the manufacturing variances can't be measured in simulations. Similar to the work on PRNU (see section 3.6.3) which has demonstrated variations between the photodiodes in an image sensor, we expect that there would be small variations between the characteristics of each memory cell.

Manufacturing variances can be detected by measuring analog values with an ADC and then averaging and correlating the results. This is very similar to the process we use to read the data from the memory array before correlating to detect missing cycles. Hence we expect

Number of writes per cycles	Bit Error Rate	Max cell value
1	0%	83
10	0%	158
127	0.001%	941
255	0.005%	1658

Table 5.3: Bit Error Rate and Max cell value for different values of N

that the output read data from a real memory device will contain manufacturing defects that could be used to uniquely identify the memory device. While it isn't the primary aim of this system, the unforgeable requirement could be achieved by measuring manufacturing defects.

Non-ideal components

The simulation assumes ideal analog memory cells that are linear, loss-less and have no limits on maximum value. Real world analog memory cells are likely to have several non-ideal characteristics which may affect the performance of this system. The non-ideal characteristics will also very greatly depending on the technology used to create the analog memory cells, for example, memristors will behave differently to capacitors.

The impact of non-ideal behavior on this system will vary greatly depending on exactly what the behavior is. For example, non-linear behavior can likely be accounted for within the control circuit whereas degradation of the values stored in memory will be a major limitation if it degrades too quickly. To further explore the impact of non-ideal behavior, we need to focus on a specific real memory cell technology.

5.5 Conclusion

Analog memory cells are memory cells that can store any value on the continuous scale between the minimum and maximum values. This is distinct from digital memory cells which can only be in one of two states (binary '0' or '1'). Unlike digital memory, analog memory is not perfect, meaning the value you read out will be slightly different to the value written to the cell. While the perfect nature of digital memory is good for reliability and data quality, it can be problematic for forensics which can use small imperfections in the data to infer details about the data's source and history.

There are a number of components that can be used to create analog memory cells such as capacitors, memristors, floating gate FETs and magnetic tape. However we mostly worked with an idealized generic analog memory cell in this chapter.

We explored how analog memory could be used to create a digital evidence bag, focusing on the requirements of tamper-evident and unforgable. By storing the digital data in the analog memory we introduce and exploit the imperfections of the analog cells to achieve the DEB requirements.

Analog memory can be designed as additive memory where instead of overwriting data, writing to the memory causes the new data to be added to the existing data in the memory. This helps prevent data been lost due to accidental (or malicious) overwriting of data. Before writing, the data is spread with orthogonal codes in the same way as used in CDMA radio communications which allows the individual pieces of data to be recoverable even after being added together.

Reading the analog memory is a destructive process where the destructiveness is often minimized as much as possible so the memory can be read many times with little loss of quality. However we used the destructive read to also erase and reset the memory cells hence making unauthorized reads harder to conceal. To detect unauthorized write, read and erase cycles, we propose chaining each sequential set of read data together by leaving a small residue value behind during the read operation. This residue value causes there to be a small correlation between sequential sets of read data. A missing correlation here suggests that an unauthorized read and erase has occurred.

The residue value can come from a number of sources but needs to be unpredictable, unique, uncorrelated and preferably unclonable. Likely this will involve a random number generator of some sort. We also need to ensure there are no correlations between the data itself so PN codes, which are deterministic but still pass the tests for statistical randomness, are used to encode the data before it is written to the memory.

Simulations of this system showed that data can be written and recovered (within an acceptable error rate) and that there was a detectable correlation between sequential sets of read data. The best results were achieved when the maximum of the residue range was set to approximately 4-5 times the step size. However a problem occurs when there are few writes per cycle (small values of N) that cause small correlations to occur in non-sequential sets of read data. This effectively means that a unauthorized read operation could go undetected if it has a small number of writes in it. This could be fixed by enforcing a large enough number of writes per cycle, effectively writing dummy data to the memory if not enough actual data is required to be written.

Overall this system appears to meet the tamper-evident requirement of a DEB, and with the right design choices, it should also be unforgeable. The offline requirement is also met since no network access is needed for this memory device to work.

5.5.1 Future Work

Adding a digital fingerprint component to this design would be simple to do since it just requires further analysis of the read data. It would provide an excellent method to ensure the DEB is unforgeable. However to test it, we need to experiment on real analog memory cells since we can't reliably simulate manufacturing variations. Experimenting with real analog memory cells would also allow the impact of any non-ideal behaviors to be explored.

In order to be sure that the tamper-evident requirement is met, further security analysis is required. There is no obvious way to tamper with the data in the analog memory, however there could be unusual use cases and edge cases that cause unexpected things to happen. Finding, and fixing these issues requires a lot of independent review similar to the process of finding security bugs in software.

Chapter 6

Memristors

6.1 Introduction

Memristors are theoretical two-terminal electronic components that link electric charge and magnetic flux. The theory behind memristors was first described in 1971 [47] and since then there has been significant practical development [53] [54]. Memristors behave like resistors but with a resistance that changes over time depending on the current through the device. When there is no current, the resistance remains unchanged which gives memristors a memory feature since their resistance can be set and then remembered over time. It is this memory feature that gives them their name, a portmanteau of "memory resistor".

While there are a number of possible applications for memristors, most research seems to be aimed at using them as an alternative to floating gate FETs in digital memory devices. This is due to the low power consumption and small physical size of the memristors along with the huge demand for digital memory [54].

In this chapter we explore the use of memristors as analog memory cells. These may provide a practical implementation of the additive memory system described in chapter 5. We started by using models to simulate the memristors and how they should function and then purchased some physical memristors and carried out tests to determine how well they fit the models. The memristors we used were experimental tungsten memristors produced by Knowm Inc. [55].

6.2 Ideal Memristors

The three fundamental circuit elements, resistors, capacitors and inductors, each link two of the four electrical properties together. As shown in table 6.1, resistors relate voltage and current, capacitors relate voltage and electric charge and inductors relate current and flux linkage. The missing element is the one that relates electric charge with flux linkage which is where Chua [47] theorized that memristors fit in.

Device	Equation	Characteristic property
Resistor	$R = \frac{dV}{dI}$	Resistance, in Ohms (Ω)
Capacitor	$C = \frac{dq}{dV}$	Capacitance, in Farads (F)
Inductor	$L = \frac{d\Phi}{dI}$	Inductance, in Henrys (H)
Memristor	$M = \frac{d\Phi}{dq}$	Memristance, in Ohms (Ω)

Table 6.1: Fundamental circuit elements and their defining equations

In addition to the relations in table 6.1, current is defined to be the time derivative of electric charge and voltage is defined to be time derivative of flux linkage.

$$I = \frac{dq}{dt}$$

$$V = \frac{d\Phi}{dt}$$

By taking the memristor equation (6.1) and substituting charge and flux linkage with their time derivatives we get (6.2)

$$M(q) = \frac{d\Phi}{dq} \tag{6.1}$$

$$M(q(t)) = \frac{d\Phi/dt}{dq/dt} = \frac{V(t)}{I(t)} \tag{6.2}$$

There is a noticeable similarity between (6.2) and Ohm's Law. That is, in an instantaneous sense, the memristor behaves like a resistor with resistance M. However unlike a resistor, M is a variable that depends on the electric charge and changes over time. $M(q(t))$ is called the memristance function and it describes how the resistance changes over time.

Since current is defined to be the time derivative of electric charge, the inverse is that electric charge is the time integral of current (6.3).

$$q(t) = \int_{-\infty}^{t_0} I dt \tag{6.3}$$

By combining the memristance function with equation (6.3), we get (6.4).

$$M(q(t)) = M(\int_{-\infty}^{t_0} I dt) \tag{6.4}$$

Equation (6.4), the memristance function, explains how the instantaneous resistance of the memristor depends on the past current through the device, and that if there is no current the resistance remains constant.

6.2.1 I-V curves

Current-voltage (I-V) curve graphs are a useful tool for understanding how all electronic components operate. For memristors, the I-V curve is the ‘pinched hysteresis’ effect curve as shown in figure 6.1. The ‘bow-tie’ shape is considered to be a defining characteristic of a memristor [56].

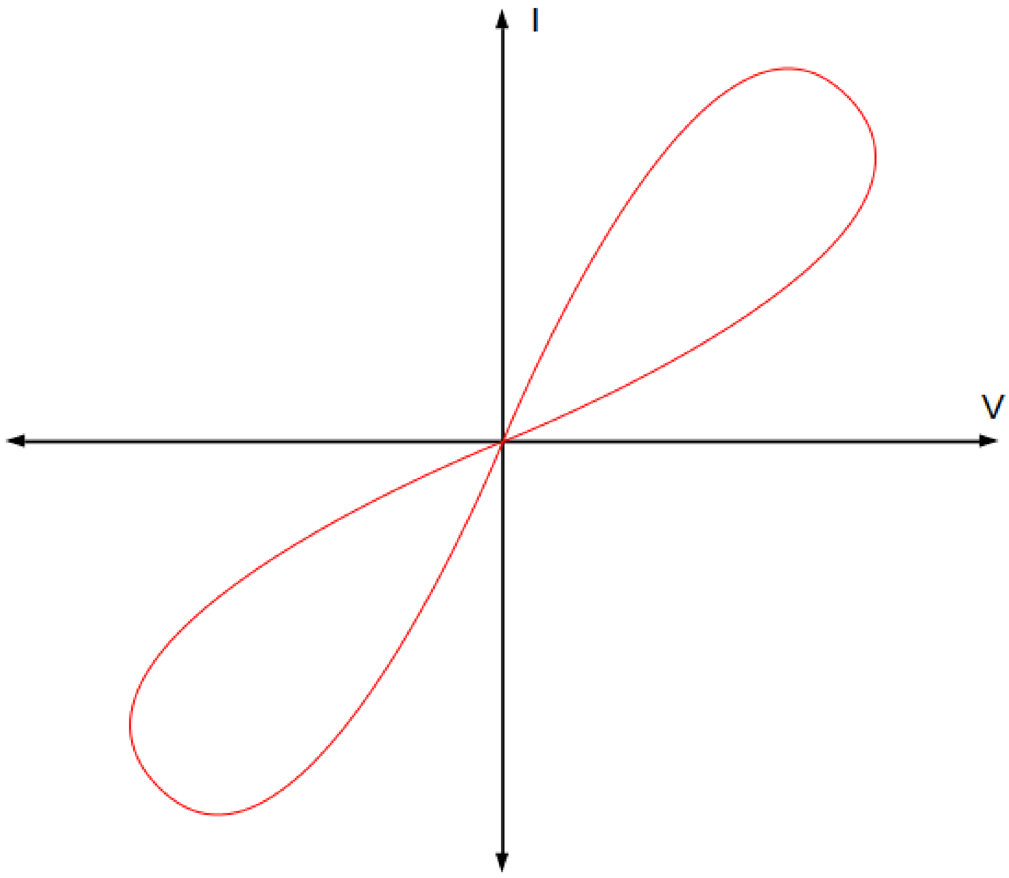


Figure 6.1: I-V curve of an ideal memristor

With resistors, the slope of their I-V curve (or line) represents their resistance (lower resistance results in a steeper I-V curve, or line). In the memristor IV curve, we can see two distinct slopes (through the origin) which represent the memristor’s resistance changing between high and low states.

The area enclosed within the I-V curve is a hysteresis effect, similar to that observed in ferromagnetic materials. This highlights the memory component of the memristor, which is that the present characteristics of the memristor depends on its past history.

6.2.2 Memristor applications

While not yet in widespread commercial use, the most promising application of memristors is in the construction of non-volatile, random access, solid state digital memory. This kind of digital memory is used in many devices including USB memory drives, solid state hard drives and SD cards and is primarily made using floating gate FETs. In comparison to floating gate FETs, memristors may offer greater memory densities, reduced power consumption and quicker performance [57].

For use as non-volatile digital memory, the memristor needs to maintain its resistance in one of two states for an extended period of time without electrical power. The high and low resistance states of the memristor can be used to represent binary '0' and '1' in digital memory.

Numerous other applications of memristors have been proposed including analog memory cells, analog signal processing and many computational applications. The required performance of the memristors varies greatly depending on the application [54].

Memristors in the digital evidence bag

Chapter 5 discussed the use of additive analog memory and its application in creating a digital evidence bag. Here we consider the use of memristors as a practical realization of the analog memory cells. The digital evidence bag system requires analog memory cells with additive write and destructive read operations.

The instantaneous resistance of the memristor represents the value stored in the memory cell. A write operation (changing/setting the resistance) is done by passing a known current through the memristor for a given time period. This is already an additive operation since the resistance value after the write depends on what it was before the write.

Reading the memrister involves measuring the instantaneous resistance. There are a number of methods to do this however they all involve passing a current through the memristor which will affect its value, meaning read operations will be destructive. A minimally destructive read can be achieved by applying a known voltage and measuring the current (or vice-versa) and then using Ohm's Law to calculate the resistance. To achieve a fully (or partially) destructive read we can apply a known current and measure the time taken for the voltage to reach a set trigger point (similar to the circuit from figure 5.5).

The circuit diagram in figure 6.2 shows how a memristor could be used to create an analog memory cell with additive write and destructive read.

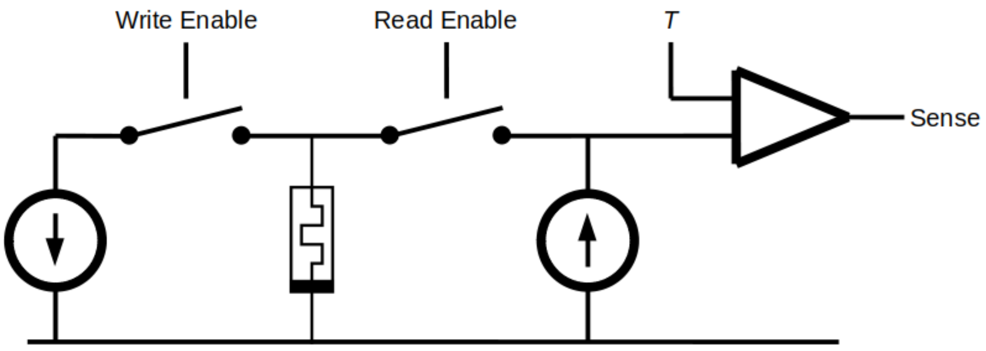


Figure 6.2: Memristor memory cell circuit with partially destructive read

6.3 Practical Memristors

6.3.1 Metastable Switch Model

The Metastable Switch (MSS) model is a more refined model that can be used to simulate the behavior of memristors [58]. Unlike the ideal model, the MSS model is designed to match the characteristics of the physical memristors. In addition to the usual deterministic properties, the MSS model is also designed to include a stochastic component which models the parts of the memristor that behave in a random way. By using the MSS model in simulations we are able to more accurately predict how the physical memristors will behave.

The MSS model is made up of two components, the memory-dependent component I_m, and the Schottky diode current I_s which models the stochastic properties of the device. The total current through the device is then given by:

$$I = \phi I_m(V, t) + (1 - \phi) I_s(V)$$

ϕ is between 0 and 1 inclusive, where $\phi = 1$ represents a device without Schottky diode effects. The manufacturer of the tungsten memristors that we experimented with suggest using $\phi = 1$, meaning that these memristors do not have any significant Schottky diode effects [8] (tungsten memristors are part of the Ag-chalcogenide family).

The memory component of the memristor is modeled as a series of N channels, each of which can be in one of two states, called A and B. Each state has a set resistance value, and the total resistance of the device is found by adding the number of channels in each state. This is done using conductance (the inverse of resistance) as shown below:

$$G_m = N_A G_A + N_B G_B$$

$$\text{(note that: } N = N_A + N_B\text{)}$$

Where N_A, N_B are the number of channels in the A and B states respectively at the given moment, and G_A, G_B and the intrinsic conductances of each of the states (which are a constant property of the given memristor).

Given the voltage across the memristor (V_m), we can then use Ohm's Law to find:

$$I_m = V_m G_m$$

The rate at which channels move between the A and B states depends on the voltage (V_m) applied to the memristor. P_A represents the probability that a single channel will transition from the B state to the A state. And likewise, P_B represents the probability that a single channel will transition from the A state to the B state. The following equations are derived in [58] to calculate these probabilities.

$$P_A = \alpha \frac{1}{1 + e^{\beta(V_m - V_A)}}$$

$$P_B = \alpha(1 - \frac{1}{1 + e^{\beta(V_m + V_B)}})$$

Where $\beta = \frac{q}{kT}$ is the thermal voltage, approximately $26mV^{-1}$ at 300 Kelvins. The two states are separated by a potential energy barrier as shown in figure 6.3. The size of this barrier is represented as V_A and V_B which are constant properties of the memristor. $\alpha = \frac{\Delta t}{t_c}$ represents the ratio between the time step period and t_c, the characteristic time scale of the memristor.

To use the MSS model in simulations, we first pick a small time step period Δt during which we assume the circuit state is constant. During each step we calculate the new resistance of the memristor and update the voltages/currents in the circuit accordingly. This process then repeats for as many time steps as needed. Smaller time steps give more accurate results at the expense of increased computational requirements.

For each step, we first calculate P_A and P_B using the present voltage across the memristor. The probabilities are then used to update N_A and N_B from the present state. And finally the conductance is calculated which can then be used to find the instantaneous current via Ohm's Law.

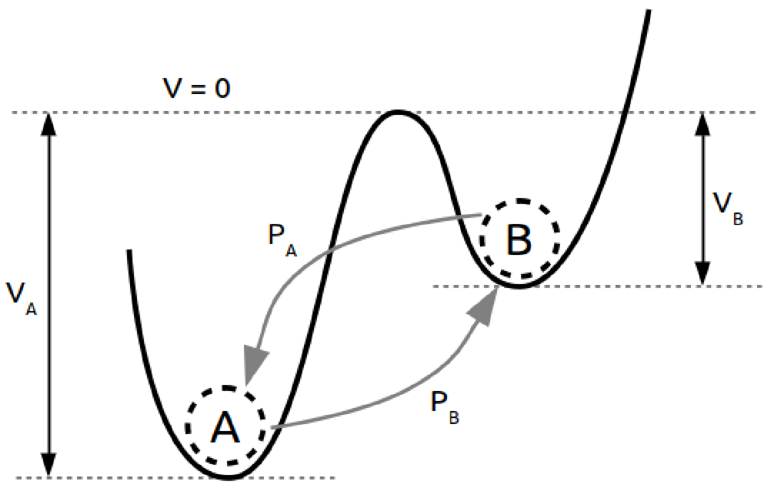


Figure 6.3: Potential energy barrier between the two states of the MSS model [8]

6.3.2 Simulations

The properties of the memristor used in the simulations were the ones suggested by the manufacturer:

$$N = 1,000,000$$

$$V_A = 400mV$$

$$V_B = 300mV$$

$$G_{OFF} = 1\mu S$$

$$G_{ON} = 3mS$$

$$t_c = 60\mu s$$

G_{OFF} and G_{ON} represent the limits of the memristance value and are used to derive G_A and G_B within the simulation.

$$G_A = \frac{G_{OFF}}{N}$$

$$G_B = \frac{G_{ON}}{N}$$

The terms of resistance, this makes the maximum $\frac{1}{G_{OFF}} = 1M\Omega$ and the minimum $\frac{1}{G_{ON}} = 333\Omega$.

Series Resistor

The test circuit used for the simulations is shown in figure 6.4. While the series resistor is not needed in the simulation, it serves two purposes when testing physical memristors. Firstly, we need a method to measure the current through the memristor and the resistor can be used as a current sense resistor. Secondly, we need to limit the current through the memristor when the memristance is in the low resistance state to prevent the memristor from overheating. In order to better simulate the real word circuit, we also included the series resistor in the simulations. The value used for the simulations was $1k\Omega$.

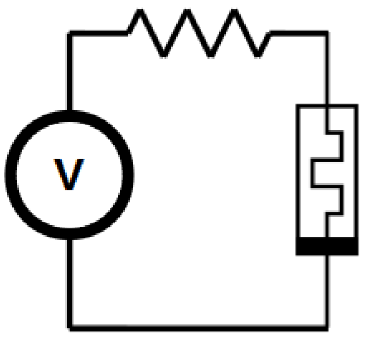


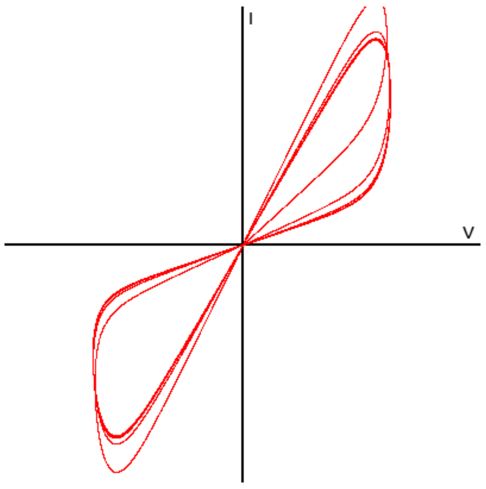
Figure 6.4: Memristor test circuit with a current limiting resistor

Sine wave response

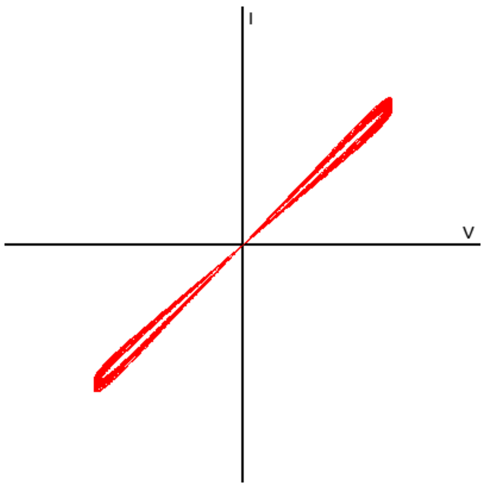
Figure 6.5a shows the I-V graph of the memristor when a 200mV sine wave at 20Hz is applied to the test circuit. The characteristic 'bow-tie' shape of memristors can clearly be seen. Figure 6.5b shows the effect of increasing the frequency to 200Hz. It can be seen that the curve tends towards the straight line similar to a resistor I-V curve as the frequency increases. This reflects the fact that the memristance doesn't have time to change as much at higher frequencies.

Initial conditions

The simulation allows us to set an initial memristance value for the memristor which allows some transient effects to be observed before a steady state is reached. Figure 6.6 shows the simulated response when the memristor's initial resistance is $1k\Omega$. It can be seen that a steady state is reached after approximately $600ms$ where the resistance varies between $10k\Omega$ and $50k\Omega$. Note that these values are not the at the limits of the memristance range.



(a) Simulated IV curve of a memristor with 20Hz sine wave



(b) Simulated IV curve of a memristor with 200Hz sine wave

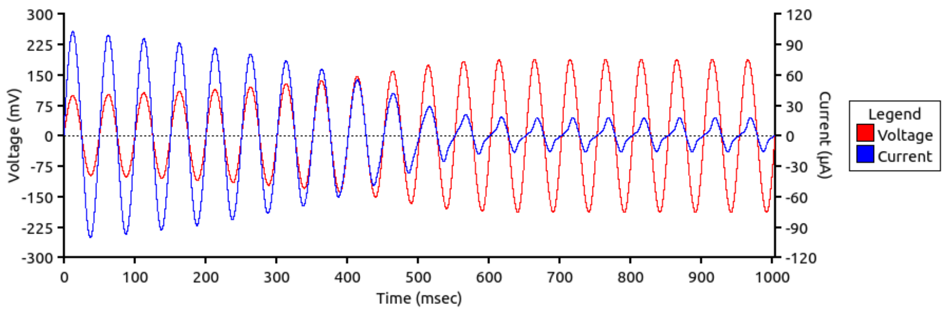
Pulse response

When using the memristor as an analog memory cell, it is likely that a pulse will be applied to the memristor to change the resistance (and hence value) stored in the memristor. Hence the simulated response to both positive and negative pulses is shown in figure 6.7 where 20ms pulses are applied to the memristor. There are a number of non-linear effects visible in this graph. The pulses are all identical in duration and magnitude however it can be seen the change in resistance due to each pulse varies significantly depending on the memristor's prior state.

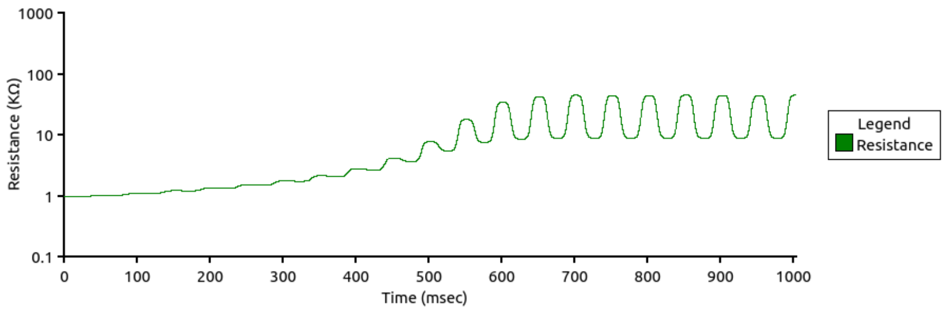
The non-linear effects are due to a number of things, mainly that the current (and hence the rate the memristance changes at) depends on the present memristance of the memristor. The series resistor in the test circuit also causes some variation in the memristor terminal voltage which affects the current. There is also a small difference between P_A and P_B which causes the memristance to move faster in one direction than the other.

Using a constant-current supply (instead of the constant-voltage supply) would remove some of these non-linear effects, however care needs to be taken on real physical memristors not to accidentally exceed the maximum power dissipation rating of the memristor. Like resistors, the power dissipation of memristors is given by $P = I^2R$ however R will of course vary with a memristor. In practical constant-current supplies there is a limit on the maximum output voltage, which if set correctly, can prevent damage to the memristor.

Alternatively, the size or duration of the pulses can be adjusted depending on the present memristance of the memristor, which would cause the memristance to move by a pre-determined amount. A feedback circuit could also be used to monitor the current and end the pulse once the current has reached a given value.



(a) Current and voltage for the memsistor



(b) Instantaneous resistance of the memristor

Figure 6.6: Simulated time-domain response of a memristor with 20Hz sine wave

Zero voltage

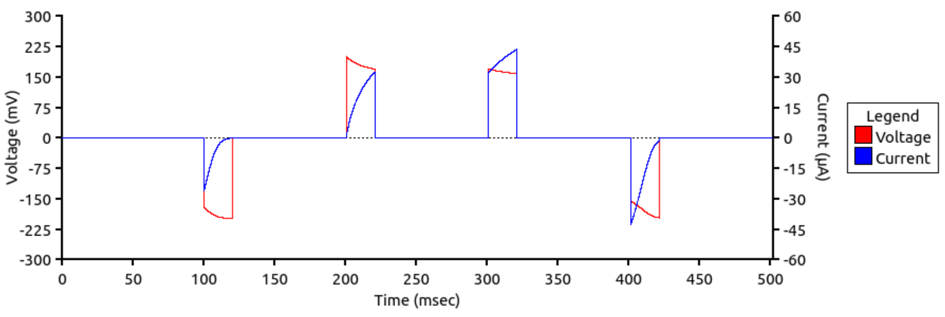
When there is no voltage applied to the memristor, there will be no current through the memristor, and hence the resistance should remain constant. However it was observed in the simulations that the resistance slowly drifts when there is a zero voltage applied. Figure 6.8 shows the result of running the simulation for 100 seconds, where it appears the resistance drifts towards a steady state of approximately $10k\Omega$.

The cause of this appears to be a flaw within the MSS model. Referring back to the equations for P_A and P_B, there is no input voltage V_m, that results in a zero probability of state change. Hence the memristor state is always changing (even if only very slowly).

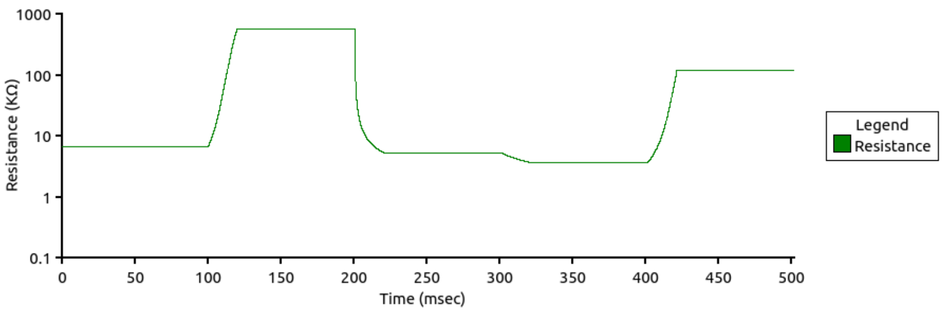
We would not expect to observe this behavior in real-world memristors since this would make them unsuitable for any applications where memory greater than a few seconds is required.

6.4 Experimental Measurements

We then tested some real tungsten memristors purchased from Knowm Inc. [55]. The circuit in figure 6.4 was implemented, with a signal generator to supply a sine wave voltage. A



(a) Current and voltage for the memsistor



(b) Instantaneous resistance of the memristor

Figure 6.7: Simulated time-domain response of a memristor to 20ms pulses

digital CRO was then used to measure the voltages across the memristor and series resistor. The voltage across the series resistor was then used to calculate the memristor current.

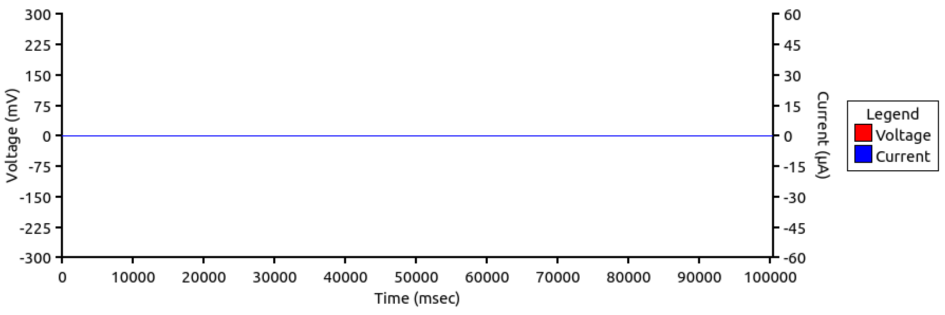
6.4.1 Sine Wave Response

Figure 6.9 shows the I-V graph when a $3Hz$ sine wave was applied to the memristor. The characteristic 'bow-tie' shape can be seen however it is a little distorted. The high and low resistance states can easily been seen but the transition between them does not happen as expected from the theory and simulations.

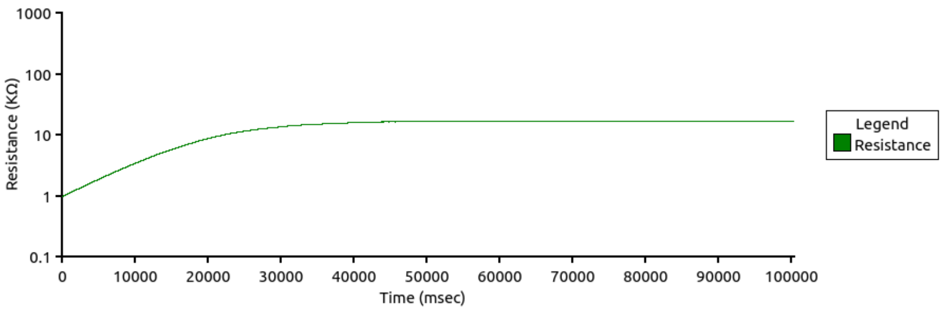
By looking at the memristor current and voltage in the time domain as shown in figure 6.10, we can see that the memristace transitions between the high the low states quickly. The vertical sections of the current waveform (in blue) show this sudden change between the high and low states.

6.4.2 Step Response

It was challenging to achieve repeatable results when running tests on the real memristors. There are some factors that we don't have complete control over such as temperature and the



(a) Current and voltage for the memsistor



(b) Instantaneous resistance of the memristor

Figure 6.8: Simulated time-domain response of a memristor with zero voltage applied

initial conditions which may have affected the tests. The memristors are also experimental components that are not yet in commercial use hence their reliability and behavior is not completely understood. We ran the step response experiment several times and the most consistent result is discussed below.

Figure 6.11 shows the step responses and it can be seen that a positive voltage causes the memristance to decrease to the minimum and that a negative voltage causes the memristance to increase to the maximum. Note that the change between states happens very quickly, in the order of about $100\mu s$ which is significantly faster than the simulations suggest $(10 - 100ms)$. The supply voltage we used was slightly higher than that recommended by the manufacturer $(1V)$. When testing with supply voltages below $1V$, we found that the memristance did not change at all. The increased supply voltage would explain why the transition between states was quicker than expected.

6.4.3 Analog Memory

To use the memristors as analog memory, we need to be able to set the memristance at any point between the high and low points, and then have it remain close to that state for a extended time period. We were unable to make this happen in our experiments. Generally

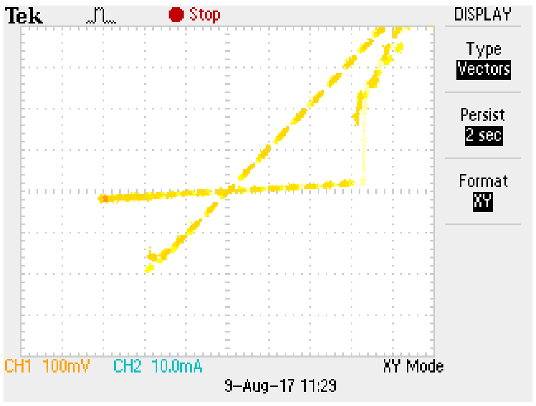


Figure 6.9: Measured memristor I-V curve with a 3Hz sine wave

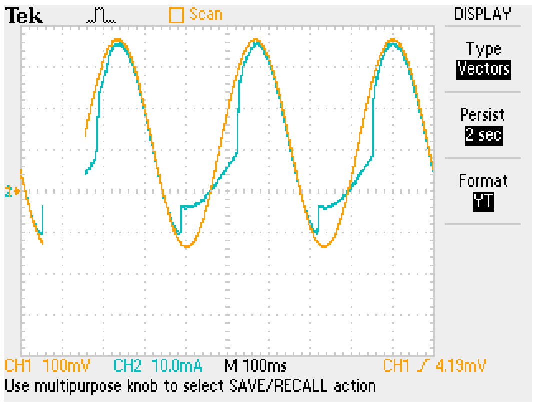


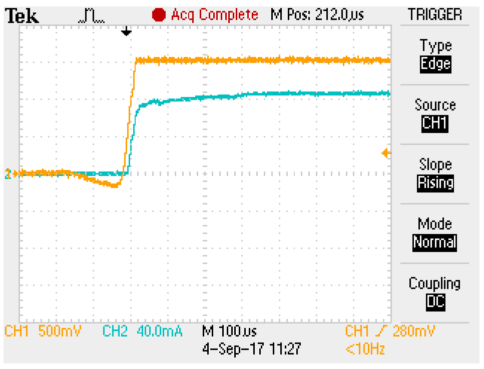
Figure 6.10: Measured memristor voltage and current with a 3Hz sine wave

we found that the memristance would change to either the saturated high or low state if power was removed while it was between those two states. However, we stress that there was very little repeatably in these experiments and further work is needed refining the experiments to better test and understand how these devices operate.

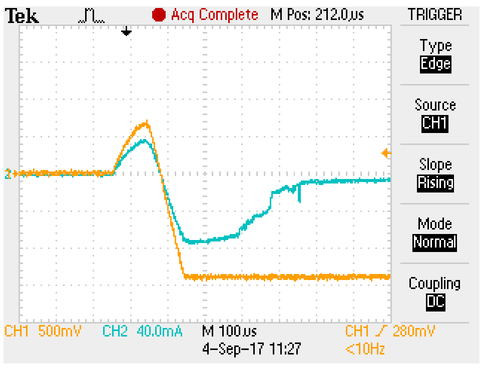
6.5 Conclusion

Memristors are circuit components that can be used to create memory. While first theorized many years ago, creating real world memristors is still an active area of research. It is also a promising area of research with many possible applications being proposed for memristors. Largely this is focused on using memristors as a better alternative to floating gate MOSFETs in digital memory applications.

We explored using memristors as the analog memory cells within the system described



(a) Positive step response



(b) Negative step response

Figure 6.11: Measured step responses of the memristor

in chapter 5 to create a digital evidence bag. The value in the analog memory cell is represented by the resistance of the memrister which can be set by passing current through the memristor. The requirements of additive write and destructive read can be achieved with a simple control circuit around the memristor memory element.

Simulations with the metastable switch model, a more refined model designed to represent the characteristics of real world memristors, suggested that these memristors could be used as analog memory. There is some non-linear behavior with the simulated memristor which needs to be accounted for when using it as analog memory.

Experiments with real tungsten memristors gave mixed results. The basic function of a memristor where the memristance changes with current was easily observed along with the (roughly) 'bow-tie' shaped I-V curve. However it appears the memristor is almost always in either the high or low memristance state and we were unable to get it to stay in any intermediate state with any sort of repeatably or reliability. The transition between the high and low states was also unpredictable and didn't always follow what the simulations suggested. One possible reason for this behavior is that practical memristors are designed for digital memory applications where the behavior between the high and low states is intended to have minimal impact on digital memory performance.

6.5.1 Future Work

Our simulations suggest that memristors can be used as analog memory cells however in practice this requires a real world memristor that functions similarly to the simulation. More work needs to be done to fully understand how real memristors can operate. If some of the seemingly erratic components of the transitions between the high and low states of the memristors were better understood, it may be possible to refine the experiment and circuit design to get more consistent results.

There are also other memristor designs that may offer better performance for analog memory applications. Our experiments were limited to tungsten memristors since they were the only practical models available.

Chapter 7

Conclusion

In this chapter we will summarize the work done in this thesis, explain the conclusions reached and highlight the novel contributions our work presented. We also outline some of the opportunities for future research in this area.

7.1 Digital forensics and the Digital Evidence Bag

Digital evidence is electronic data that is useful for establishing facts around legal problems. When handling any evidence, it is important the evidence integrity and chain of custody is maintained for the entire forensic process. This includes the identification, collection, examination, analysis and presentation phases of the forensic process. The procedures used to ensure evidence integrity and chain of custody are designed for traditional physical evidence and doesn't always apply that well to digital evidence.

After reviewing a number of standards for digital evidence handling in the various stages of the forensic process, we introduced the concept of a Digital Evidence Bag. This is an electronic device for the transport and handling of digital evidence which like a traditional evidence bag, it is designed to maintain evidence integrity and chain of custody.

By researching the requirements of traditional evidence bags, we then adapted them based on the standards for digital evidence handling to create the requirements for a DEB. The four requirements being tamper evident, unforgeable, clean and offline. Tamper evident prevents accidental and malicious alterations to the data going undetected. Unforgeable prevents fake copies of the DEB being created. Clean ensures that any data stored on the DEB is not contaminated by data from previous uses. And finally, the offline requirement is needed since evidence may be collected from places without a network connection.

7.2 Information security technologies

We reviewed many of the current systems used in information security and looked at the underlying security concepts of each. We found that there where four main approaches used: cryptography, secure hardware, widely witnessed (blockchain) and exploiting manufacturing defects/variances.

The most commonly used of these, cryptography, relies on a secret key being kept secret. There are also issues around how to establish trust in given keys when using them for authentication purposes. Secure hardware is also quite common and involves designing hardware (in particular ICs) such that they do not reveal, or permit the modification of, sensitive data. This ranges from simple write-once memory up to specialist hardware designed for the secure storage of cryptographic keys.

Blockchains are based on the concept that once some data has been witnessed by a large number of people, it can not be changed without someone noticing. While this is a simple concept, blockchain is a relatively new technology that has allowed the widely witnessed concept to operate efficiently on a commercial scale. Finally the concept of exploiting manufacturing defects and variances for security purposes is still quite new and is largely in the research and development phase. However it does appear to offer some very novel and elegant solutions to the aims of information security.

7.3 Fingerprinting digital memory

We explored how to create a digital fingerprint for conventional digital memory (based on floating gate FETs). Since the DEB is likely to contain digital memory components, a digital fingerprint could be used to meet the unforgeable requirement of the DEB. While there are a number of parameters within digital memory devices that can be measured to create a digital fingerprint, we choose to look at measuring the analog output voltage from the memory cells. When compared to the existing research in this area, measuring the analog output voltages could offer a quick, reliable and non-destructive (to the data in the memory) method of fingerprinting digital memory.

The experiments we carried out were designed to measure the analog output voltage from the digital memory cells. However we found that the digital output buffers within the memory chips effectively masked any variations in the output voltages from the memory cells themselves. As a result, we were actually measuring the variations of the output buffers rather than the memory cells. These variations in the output buffers could be used as part of a digital fingerprint however there are only a small number of them (8 for the chips we tested) which is not enough to ensure a unique fingerprint on their own.

We have demonstrated that the variations in output voltage from the memory cells can not be easily measured in off-the-shelf digital memory chips. However a custom designed memory chip that allows the output of the memory cells to be sampled directly by an ADC without any digital circuit elements along the way may offer a method to create a digital fingerprint from the variations in the memory cell output voltages.

7.4 Creating an analog memory based DEB

We then considered the use of analog memory cells to create the DEB and looked at how this could be achieved. The additive design of the analog memory cells means that data is not overwritten, instead writing new data just results in it getting added on top of the existing data. By combining this with a destructive read process that is designed to make unauthorized erases/resets to the memory detectable, we created a reusable, tamper-evident memory device. The unforgeable requirement can also be meet with this design.

A system of linking each cycle was proposed similar to the linking of data blocks in hashchains. This would mean unauthorized reads and erasures would create a missing link in the chain which would be detectable. A method for creating this chain within the analog memory device was developed whereby correlating the read data from sequential cycles is used to verify the existence of a link.

Simulations were then carried out to test this concept. The results of the simulation showed that data can be stored and retrieved from the analog memory with a BER that can be as low as 2.5%. With the right parameters set, read data from sequential cycles has a correlation that is at a discernible level. This suggests that any missing cycles would be detectable, meaning unauthorized erases are detectable. This should then fulfill the tamper-evident requirement of the DEB. The unforgeable requirement can likely be achieved with a digital fingerprint, however demonstrating this requires experimentation with real-world analog memory cells.

7.5 Using memristors as analog memory cells

We then explored the use of memristors to create a real-world analog memory cell. Memristors are a two terminal electronic circuit component that behaves like a resistor, but with a resistance that changes depending on the historical current through the memristor. This allows the memristor to be used a memory element where the present resistance of the memristor is used to represent the stored value.

While there has been a lot of research into creating a memristor, they are still in the development phase without any large scale commercial uses. However there have been a number of potential applications proposed, the most common one being using memristors as a smaller and more power efficient alternative to floating gate FETs in non-volatile digital memory. Experimental memristors are available which we used to explore their suitability for creating analog memory.

Initially we carried out simulations using the Metastable Switch model of the memristors which showed that in principle the memristors could be used as analog memory cells. There is however some non-linear behavior which would need to be taken into account when designing a memristor based analog memory device. We then experimented with some real-world tungsten memristors and found that they didn't match the models as well as expected. They tended to be in either the high or low memristance state and the transition between these states appeared to be unpredictable and unrepeatable. This suggested that the memristors may have been optimized for digital memory applications. However our experiments were relatively simple and further work to fully understand how these real-world memristors operate may lead to more consistent results.

7.6 Future Work

This work has introduced and outlined the need and requirements for a Digital Evidence Bag. Drawing from previous work in information security, we have then proposed some novel ideas for creating a DEB.

Fingerprinting digital memory can be used to meet the unforgeable requirement of the DEB. While we have shown that there are variations between the output voltage levels of the output buffers, further work is required to confirm that there are also variations between the output voltage levels of the memory cells. We believe this idea is worth pursuing since measuring the variations in output voltages will likely offer a simple method of digital fingerprinting memory devices in a way that is non-destructive to both the memory device and the data within it.

Exploiting the analog characteristics of memory cells also provides a novel solution to the requirements of the DEB. We have shown that by using analog memory cells in an additive write and destructive read system we can create a tamper-evident data storage device. However this has only been demonstrated in simulation hence further work is needed to demonstrate this concept with real analog memory cells. Our simulations have suggested that memristors could be a suitable component to implement analog memory cells, however other technologies should also be considered.

Appendix A

Full correlation data for individual bit lines from NMOS memory chips

		Read 1			Read 2			Read 3		
		Chip 1	Chip 2	Chip 3	Chip 1	Chip 2	Chip 3	Chip 1	Chip 2	Chip 3
Read 3	Chip 3	D0: 83.6% D1: 93.1% D2: 91.4% D3: 89.1% D4: 91.1% D5: 90.3% D6: 92.8% D7: 92.8% Avg: 90.5%	D0: 80.8% D1: 93.2% D2: 91.3% D3: 89.3% D4: 91.0% D5: 90.7% D6: 93.1% D7: 93.4% Avg: 90.5%	D0: 83.1% D1: 91.7% D2: 92.1% D3: 89.4% D4: 90.5% D5: 89.7% D6: 92.1% D7: 92.1% Avg: 90.1%	D0: 78.8% D1: 92.1% D2: 91.2% D3: 88.9% D4: 91.0% D5: 89.6% D6: 93.6% D7: 92.4% Avg: 89.7%	D0: 77.9% D1: 92.0% D2: 90.7% D3: 87.6% D4: 90.5% D5: 88.4% D6: 92.3% D7: 92.9% Avg: 89.1%	D0: 74.6% D1: 92.3% D2: 91.4% D3: 87.4% D4: 90.4% D5: 87.4% D6: 92.5% D7: 92.6% Avg: 88.6%	D0: 81.6% D1: 93.0% D2: 92.3% D3: 90.0% D4: 90.8% D5: 89.9% D6: 92.6% D7: 92.6% Avg: 90.3%	D0: 82.7% D1: 93.0% D2: 91.6% D3: 89.6% D4: 91.2% D5: 90.4% D6: 93.8% D7: 93.5% Avg: 90.7%	D0: 100% D1: 100% D2: 100% D3: 100% D4: 100% D5: 100% D6: 100% D7: 100% Avg: 100%
	Chip 2	D0: 87.8% D1: 93.3% D2: 90.9% D3: 89.0% D4: 91.8% D5: 90.3% D6: 93.0% D7: 92.5% Avg: 91.1%	D0: 84.6% D1: 92.4% D2: 91.8% D3: 89.8% D4: 91.6% D5: 90.8% D6: 93.1% D7: 92.8% Avg: 90.9%	D0: 87.0% D1: 91.6% D2: 90.9% D3: 88.8% D4: 91.5% D5: 89.7% D6: 92.9% D7: 92.4% Avg: 90.6%	D0: 83.6% D1: 91.8% D2: 91.1% D3: 89.1% D4: 91.2% D5: 90.2% D6: 94.1% D7: 93.8% Avg: 90.6%	D0: 80.9% D1: 91.6% D2: 90.6% D3: 88.0% D4: 90.9% D5: 88.5% D6: 92.5% D7: 92.4% Avg: 89.4%	D0: 79.2% D1: 92.0% D2: 90.7% D3: 87.8% D4: 91.3% D5: 88.1% D6: 93.6% D7: 92.8% Avg: 89.4%	D0: 84.8% D1: 92.9% D2: 92.1% D3: 89.6% D4: 91.6% D5: 90.3% D6: 92.9% D7: 92.7% Avg: 90.9%	D0: 100% D1: 100% D2: 100% D3: 100% D4: 100% D5: 100% D6: 100% D7: 100% Avg: 100%	
	Chip 1	D0: 85.2% D1: 93.3% D2: 91.5% D3: 89.1% D4: 91.4% D5: 90.2% D6: 93.6% D7: 93.9% Avg: 91.0%	D0: 82.4% D1: 92.6% D2: 91.6% D3: 89.6% D4: 91.2% D5: 90.6% D6: 94.1% D7: 93.9% Avg: 90.8%	D0: 85.2% D1: 92.1% D2: 92.0% D3: 90.2% D4: 91.3% D5: 90.2% D6: 92.8% D7: 93.8% Avg: 90.9%	D0: 80.5% D1: 91.8% D2: 91.1% D3: 88.6% D4: 91.0% D5: 89.2% D6: 92.6% D7: 92.6% Avg: 89.7%	D0: 77.8% D1: 90.6% D2: 89.7% D3: 86.5% D4: 90.9% D5: 87.5% D6: 92.5% D7: 92.1% Avg: 88.4%	D0: 76.8% D1: 92.4% D2: 90.9% D3: 88.0% D4: 91.0% D5: 87.9% D6: 92.2% D7: 92.4% Avg: 88.9%	D0: 100% D1: 100% D2: 100% D3: 100% D4: 100% D5: 100% D6: 100% D7: 100% Avg: 100%		

Table A.1: Correlating the NMOS data from each bit line separately (shaded cells represent correlation between data from the same memory chip) *(part 1)*

		Read 1			Read 2		
		Chip 1	Chip 2	Chip 3	Chip 1	Chip 2	Chip 3
Read 2	Chip 3	D0: 77.6% D1: 91.6% D2: 90.0% D3: 86.7% D4: 90.9% D5: 87.6% D6: 92.0% D7: 92.1% Avg: 88.6%	D0: 76.9% D1: 92.4% D2: 89.9% D3: 87.3% D4: 90.5% D5: 88.2% D6: 92.2% D7: 92.7% Avg: 88.8%	D0: 77.9% D1: 92.1% D2: 91.5% D3: 87.2% D4: 91.0% D5: 87.4% D6: 92.3% D7: 92.4% Avg: 89.0%	D0: 75.1% D1: 91.5% D2: 89.8% D3: 86.8% D4: 90.4% D5: 87.4% D6: 92.9% D7: 91.2% Avg: 88.2%	D0: 72.1% D1: 89.7% D2: 88.6% D3: 84.7% D4: 90.0% D5: 85.3% D6: 91.3% D7: 91.0% Avg: 86.6%	D0: 100% D1: 100% D2: 100% D3: 100% D4: 100% D5: 100% D6: 100% D7: 100% Avg: 100%
	Chip 2	D0: 82.8% D1: 91.2% D2: 89.7% D3: 88.3% D4: 91.1% D5: 89.5% D6: 92.8% D7: 92.4% Avg: 89.7%	D0: 78.3% D1: 92.0% D2: 89.7% D3: 87.6% D4: 91.5% D5: 88.6% D6: 93.0% D7: 92.7% Avg: 89.2%	D0: 79.3% D1: 89.2% D2: 88.8% D3: 85.9% D4: 89.7% D5: 86.8% D6: 90.8% D7: 90.9% Avg: 87.7%	D0: 77.4% D1: 91.4% D2: 89.4% D3: 86.9% D4: 91.1% D5: 88.3% D6: 92.8% D7: 92.2% Avg: 88.7%	D0: 100% D1: 100% D2: 100% D3: 100% D4: 100% D5: 100% D6: 100% D7: 100% Avg: 100%	
	Chip 1	D0: 83.3% D1: 91.8% D2: 90.3% D3: 88.5% D4: 91.2% D5: 89.8% D6: 92.7% D7: 92.4% Avg: 90.0%	D0: 80.7% D1: 93.6% D2: 90.3% D3: 88.5% D4: 91.5% D5: 90.4% D6: 93.1% D7: 92.4% Avg: 90.1%	D0: 83.8% D1: 92.7% D2: 91.0% D3: 89.1% D4: 90.5% D5: 89.8% D6: 92.3% D7: 92.0% Avg: 90.2%	D0: 100% D1: 100% D2: 100% D3: 100% D4: 100% D5: 100% D6: 100% D7: 100% Avg: 100%		

Table A.2: Correlating the NMOS data from each bit line separately (shaded cells represent correlation between data from the same memory chip) *(part 2)*

		Read 1		
		Chip 1	Chip 2	Chip 3
Read 1	Chip 3	D0: 88.0% D1: 91.7% D2: 91.6% D3: 89.1% D4: 91.4% D5: 90.0% D6: 92.6% D7: 93.6% Avg: 91.0%	D0: 83.8% D1: 92.5% D2: 91.0% D3: 89.3% D4: 90.9% D5: 90.1% D6: 92.3% D7: 93.3% Avg: 90.4%	D0: 100% D1: 100% D2: 100% D3: 100% D4: 100% D5: 100% D6: 100% D7: 100% Avg: 100%
	Chip 2	D0: 84.5% D1: 92.1% D2: 91.0% D3: 89.4% D4: 91.5% D5: 90.8% D6: 94.4% D7: 94.0% Avg: 91.0%	D0: 100% D1: 100% D2: 100% D3: 100% D4: 100% D5: 100% D6: 100% D7: 100% Avg: 100%	
	Chip 1	D0: 100% D1: 100% D2: 100% D3: 100% D4: 100% D5: 100% D6: 100% D7: 100% Avg: 100%		

Table A.3: Correlating the NMOS data from each bit line separately (shaded cells represent correlation between data from the same memory chip) *(part 3)*

Bibliography

[1] S. Mason and D. Seng, *Electronic Evidence, Fourth Edition.* University of London, 2017.

[2] A. Arnes, *Digital Forensics.* John Wiley & Sons, 2018.

[3] S. Vanstone, A. Menezes, and P. v. Oorschot, *Handbook of Applied Cryptography.* CRC Press, 1996.

[4] C. Herder, M.-D. Yu, F. Koushanfar, and S. Devadas, "Physical Unclonable Functions and Applications: A Tutorial," *Proceedings of the IEEE*, vol. 102, no. 8, pp. 1126–1141, 2014.

[5] *Datasheet: 27C64 65,536-Bit (8,192 x 8) UV Erasable CMOS PROM*, National Semiconductor.

[6] C. Maxfield, *Bebop to the Boolean boogie an unconventional guide to electronics.* Elsevier/Newnes, 2009.

[7] D. Navarro, Z. Feng, V. Viswanathan, L. Carrel, and I. O'Connor, "Image toolbox for CMOS image sensors simulations in Cadence ADE," in *International Conference on Design and Modeling in Science, Education, and Technology: DeMset*, 2011, p. 5.

[8] T. W. Molter, "The Generalized Metastable Switch Memristor Model," 2015. [Online]. Available: https://knowm.org/the-generalized-metastable-switch-memristor-model/

[9] S. Samonas and D. Coss, "The CIA strikes back: Redefining confidentiality, integrity and availability in security." *Journal of Information System Security*, vol. 10, no. 3, 2014.

[10] "Exchangeable image file format for digital still cameras: Exif Version 2.3," JEITA, Standard, 2010.

[11] *SWGDE Framework of a Quality Management System for Digital and Multimedia Evidence Forensic Science Service Providers*, Scientific Working Group on Digital

Evidence, 2017. [Online]. Available:
https://www.swgde.org/documents/Current%20Documents/SWGDE%20Framework%20of%20a%20Quality%20Management%20System%20for%20Digital%20and%20Multimedia%20Evidence%20Forensic%20Science%20Service%20Providers

[12] "Global Guidelines for Digital Forensics Laboratories," Interpol, Standard, 2019.

[13] "ETSI TS 103 643 - Techniques for assurance of digital material used in legal proceedings," ETSI, Standard, 2020.

[14] "ISO/IEC 17025 – General requirements for the competence of testing and calibration laboratories," International Organization for Standardization, Standard, 2017.

[15] National Association of Testing Authorities. [Online]. Available: https://www.nata.com.au/

[16] P. Beardmore, G. Fellows, and P. Sommer, "UK ISO 17025 Digital Forensics Survey April 2017: Results," 2017. [Online]. Available: https://docplayer.net/storage/65/52798257/1635827021/FVlLCkdy5TfWQXX8SZBjUA/52798257.pdf

[17] Microsoft Corporation, "Cryptography Tools, Introduction to Code Signing," 2018. [Online]. Available:
https://docs.microsoft.com/en-us/windows/win32/seccrypto/cryptography-tools

[18] *Australia and New Zealand Guidelines for Digital Imaging Processes*, Australian and New Zealand Policing Advisory Agency, 2013. [Online]. Available: http://www.anzpaa.org.au/ArticleDocuments/282/2013%20Australia%20and%20New%20Zealand%20Guidelines%20for%20Digital%20Imaging%20Processes.pdf.aspx

[19] T. Neville and M. Sorell, "Audit log for forensic photography," in *International Conference on Forensics in Telecommunications, Information, and Multimedia.* Springer, 2009, pp. 142–152.

[20] P. A. Blythe and J. J. Fridrich, "Secure Digital Camera," in *The Digital Forensic Research Conference*, 2004.

[21] *Scenes of Crime Examination Best Practice Manual*, ENSFI Scenes of Crime Working Group, 2012. [Online]. Available:
https://library.college.police.uk/docs/appref/ENFSI-BPM-v1_0.pdf

[22] Australian Goverment, "Evidence Act 1995," 1995, available at:
https://www.legislation.gov.au/Details/C2016C00605.

[23] J. Byers, "Interview with Brevet Sergeant Jeremy Byers, South Australia Police," October 2019.

[24] D. Davies, "A brief history of cryptography," *Information Security Technical Report*, vol. 2, no. 2, pp. 14–17, 1997.

[25] R. Shirey, "Internet Security Glossary, Version 2," RFC Editor, RFC 4949, 2007.

[26] S. Al-Kuwari, J. Davenport, and R. Bradford, "Cryptographic Hash Functions: Recent Design Trends and Security Notions." *IACR Cryptology ePrint Archive*, vol. 2011, p. 565, 01 2011.

[27] S. Debnath, A. Chattopadhyay, and S. Dutta, "Brief review on journey of secured hash algorithms," in *2017 4th International Conference on Opto-Electronics and Applied Optics (Optronix)*, 2017, pp. 1–5.

[28] Access Now, "The weakest link in the chain: Vulnerabilities in the SSL certificate authority system and what should be done about them," 2011. [Online]. Available: https://www.accessnow.org/cms/assets/uploads/archive/docs/Weakest_Link_in_the_Chain.pdf

[29] P. R. Zimmermann, *The Official PGP User's Guide.* Cambridge, MA, USA: MIT Press, 1995.

[30] S. Nakamoto, "Bitcoin: A Peer-to-Peer Electronic Cash System," 2009. [Online]. Available: https://bitcoin.org/bitcoin.pdf

[31] A. Buldas, A. Kroonmaa, and R. Laanoja, "Keyless signatures' infrastructure: How to build global distributed hash-trees," in *Nordic Conference on Secure IT Systems.* Springer, 2013, pp. 313–320.

[32] *An industrial scale blockchain*, Guardtime. [Online]. Available: https://www.guardtime.com

[33] S. P. Skorobogatov, "Semi-invasive attacks – A new approach to hardware security analysis," University of Cambridge, Computer Laboratory, Tech. Rep. UCAM-CL-TR-630, Apr. 2005. [Online]. Available: https://www.cl.cam.ac.uk/techreports/UCAM-CL-TR-630.pdf

[34] Ponemon Institute LLC, "Hardware Security Modules Global Market Study," 2014. [Online]. Available: {https://www.ponemon.org/local/upload/file/HP%20Atalla%20Report%20FINAL%202.pdf}

[35] F. Demaertelaere, "Hardware security modules," *SecAppDev.org-Secure Application Development*, 2010.

[36] P. Paul, S. Moore, and S. Tam, "Tamper Protection for Security Devices," in *2008 Bio-inspired, Learning and Intelligent Systems for Security*, 2008, pp. 92–96.

[37] S. M. Ross, *Introductory Statistics (Third Edition).* Academic Press, 2010.

[38] J. Lukas, J. Fridrich, and M. Goljan, “Digital camera identification from sensor pattern noise,” *IEEE Transactions on Information Forensics and Security*, vol. 1, no. 2, pp. 205–214, 2006.

[39] P. Prabhu, A. Akel, L. M. Grupp, W.-K. S. Yu, G. E. Suh, E. Kan, and S. Swanson, “Extracting Device Fingerprints from Flash Memory by Exploiting Physical Variations,” in *Proceedings of the 4th International Conference on Trust and Trustworthy Computing*, ser. TRUST’11. Berlin, Heidelberg: Springer-Verlag, 2011, p. 188–201.

[40] J. P. van Zandwijk, “Bit-errors as a source of forensic information in NAND-flash memory,” *Digital Investigation*, vol. 20, pp. S12–S19, 2017, dFRWS 2017 Europe. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1742287617300294

[41] R. Matthews, M. Sorell, and N. Falkner, “An analysis of optical contributions to a photo-sensor’s ballistic fingerprints,” *Digital Investigation*, vol. 28, pp. 139–145, 2019. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1742287618303372

[42] P. Hasler, B. Minch, and C. Diorio, “Floating-gate devices: they are not just for digital memories any more,” in *1999 IEEE International Symposium on Circuits and Systems (ISCAS)*, vol. 2, 1999, pp. 388–391 vol.2.

[43] *Datasheet: PIC18F2458/2553/4458/4553 28/40/44-Pin High-Performance, Enhanced Flash, USB Microcontrollers with 12-Bit A/D and nanoWatt Technology*, Microchip.

[44] D. L. Feucht, *Handbook of Analog Circuit Design.* Academic Press, 1990.

[45] K. H. Wee, T. Nozawa, T. Yonezawa, Y. Yamashita, T. Shibata, and T. Ohmi, “High-precision analog EEPROM with real-time write monitoring,” in *ISCAS 2001. The 2001 IEEE International Symposium on Circuits and Systems (Cat. No.01CH37196)*, vol. 1, 2001, pp. 105–108 vol. 1.

[46] E. R. Fossum and D. B. Hondongwa, “A Review of the Pinned Photodiode for CCD and CMOS Image Sensors,” *IEEE Journal of the Electron Devices Society*, vol. 2, no. 3, pp. 33–43, 2014.

[47] L. Chua, “Memristor-The missing circuit element,” *IEEE Transactions on Circuit Theory*, vol. 18, no. 5, pp. 507–519, 1971.

[48] J. Mallinson, "Tutorial review of magnetic recording," *Proceedings of the IEEE*, vol. 64, no. 2, pp. 196–208, 1976.

[49] P. Gutmann, "Secure Deletion of Data from Magnetic and Solid-State Memory," in *6th USENIX Security Symposium (USENIX Security 96).* San Jose, CA: USENIX Association, Jul. 1996. [Online]. Available: https://www.usenix.org/conference/6th-usenix-security-symposium/secure-deletion-data-magnetic-and-solid-state-memory

[50] A. Viterbi, *CDMA: Principles of Spread Spectrum Communication.* Addison-Wesley Publishing Company, 1995.

[51] Electronics Notes, "CDMA Orthogonal Spreading Codes," accessed: 2021-08-23. [Online]. Available: https://www.electronics-notes.com/articles/radio/dsss/cdma-orthogonal-spreading-codes.php

[52] M. Turan, A. Doganaksoy, and S. Boztas, "On independence and sensitivity of statistical randomness tests," in *Sequences and their Applications - SETA 2008.* Germany: Springer, 2008, pp. 18–29.

[53] R. S. Williams, "How We Found The Missing Memristor," *IEEE Spectrum*, vol. 45, no. 12, pp. 28–35, 2008.

[54] T. Prodromakis and C. Toumazou, "A review on memristive devices and applications," in *2010 17th IEEE International Conference on Electronics, Circuits and Systems*, 2010, pp. 934–937.

[55] *Datasheet: Knowm Self Directed Channel Memristors*, Knowm Inc. [Online]. Available: https://knowm.org/downloads/Knowm_Memristors.pdf

[56] L. Chua and S. M. Kang, "Memristive devices and systems," *Proceedings of the IEEE*, vol. 64, no. 2, pp. 209–223, 1976.

[57] R. Marani, G. Gelao, and A. G. Perri, "A review on memristor applications," *arXiv preprint arXiv:1506.06899*, 2015.

[58] T. W. Molter and M. A. Nugent, "The Generalized Metastable Switch Memristor Model," in *CNNA 2016; 15th International Workshop on Cellular Nanoscale Networks and their Applications*, 2016, pp. 1–2.